Universidad Autónoma de Madrid

Ph.D. Thesis

Ensemble learning in the presence of noise

Author:
Maryam Sabzevari

Supervisors:
Dr. Gonzalo Martínez Muñoz,
Dr. Alberto Suárez González

Computer Science Department
Escuela Politécnica Superior
Universidad Autónoma de Madrid
C/ Francisco Tomás y Valiente, 11
Madrid 28049 Spain.

January 2019

Abstract

The availability of vast amounts of data from a variety sources greatly expands the possibilities for an intelligent exploitation of the information contained therein. Notwithstanding, the extraction of knowledge from the raw data is a challenging task that requires the development of learning methods that are effective, efficient, and robust. One of the main difficulties in learning by induction from data is contamination by noise. In this thesis, we address the problem of automatic learning in the presence of noise. For this purpose, the ensemble learning paradigm is used. Our goal is to build collections of base learners whose outputs are combined so as to improve not only the accuracy but also the robustness of the predictions.

A first contribution of this thesis is to take advantage of subsampling to build bootstrap ensembles (e.g., bagging, or random forest) that are both accurate and robust. This idea of using subsampling as a regularization mechanism is also exploited for the detection of noisy instances.

Another contribution of this thesis is vote-boosting, a sequential ensemble learning method especially designed to be robust to noise in the class labels. Vote-boosting partially remedies the excessive sensitivity to this type of noise of standard boosting algorithms, such as AdaBoost. In standard boosting, the distribution of weights in the training data is progressively modified to emphasize misclassified instances.By contrast, in vote boosting the emphasis is based on the level of uncertainty (agreement or disagreement) of the ensemble prediction, irrespective of the class label. Similar to boosting, vote-boosting can be analyzed as a gradient descent optimization in a functional space.

One of the open problems in ensemble learning is how to build ensembles of strong classifiers. The main difficulty is achieving diversity of the base learners without a significant deterioration of the performance, and without incurring high computational costs. In this thesis we propose to build ensembles of SVMs with the help of both randomization and optimization mechanisms. Thanks to this combination of complementary strategies it is possible to build ensembles of SVMs that are much faster to train and are potentially more accurate than a single, fully optimized SVM.

In a final contribution, we develop a procedure to build heterogeneous ensembles that interpolate between homogeneous ensembles composed of different types of learners. The optimal composition of the ensemble is determined through cross-validation.

Resumen

La disponibilidad de grandes cantidades de datos provenientes de diversas fuentes amplía enormemente las posibilidades para una explotación inteligente de la información. No obstante, la extracción de conocimiento a partir de datos en bruto es una tarea compleja que requiere el desarrollo de métodos de aprendizaje eficientes y robustos. Una de las principales dificultades en el aprendizaje automático es la presencia de ruido en los datos. En esta tesis, abordamos el problema del aprendizaje automático en presencia de ruido. Para este propósito, nos centraremos en el uso de conjuntos de clasificadores. Nuestro objetivo es crear colecciones de aprendices base cuyos resultados, al ser combinados, mejoren no solo la precisión sino también la robustez de las predicciones.

Una primera contribución de esta tesis es aprovechar el ratio de submuestreo para construir conjuntos de clasificadores basados en bootstrap (como bagging o random forests) precisos y robustos. La idea de utilizar el submuestreo como mecanismo de regularización también se explota para la detección de ejemplos ruidosos. En concreto, los ejemplos que están mal clasificados por una fracción de los miembros del conjunto se marcan como ruido. El valor óptimo de este umbral se determina mediante validación cruzada. Las instancias ruidosas se eliminan (filtrado) o se corrigen sus etiquetas de su clase (limpieza). Finalmente, se construye un conjunto de clasificadores utilizando los datos de entrenamiento limpios (filtrados o limpiados).

Otra contribución de esta tesis es vote-boosting, un método de conjuntos secuencial especialmente diseñado para ser robusto al ruido en las etiquetas de clase. Vote-boosting reduce la excesiva sensibilidad a este tipo de ruido de los algoritmos basados en boosting, como adaboost. En general, los algoritmos basados en booting modifican la distribución de pesos en los datos de entrenamiento progresivamente para enfatizar instancias mal clasificadas. Este enfoque codicioso puede terminar dando un peso excesivamente alto a instancias cuya etiqueta de clase sea incorrecta. Por el contrario, en vote-boosting, el énfasis se basa en el nivel de incertidumbre (acuerdo o desacuerdo) de la predicción del conjunto, independientemente de la etiqueta de clase. Al igual que en boosting, vote-boosting se puede analizar como una optimización de descenso por gradiente en espacio funcional.

Uno de los problemas abiertos en el aprendizaje de conjuntos es cómo construir combinaciones de clasificadores fuertes. La principal dificultad es lograr diversidad entre los clasificadores base sin un deterioro significativo de su rendimiento y sin aumentar en exceso el coste computacional. En esta tesis, proponemos construir conjuntos de SVM con la ayuda de mecanismos de aleatorización y optimización. Gracias a esta

combinación de estrategias complementarias, es posible crear conjuntos de SVM que son mucho más rápidos de entrenar y son potencialmente más precisos que un SVM individual optimizado.

Por último, hemos desarrollado un procedimiento para construir conjuntos heterogéneos que interpolan sus decisiones a partir de conjuntos homogéneos compuestos por diferentes tipos de clasificadores. La composición óptima del conjunto se determina mediante validación cruzada.

Acknowledgements

First and foremost, I would like to express my sincere thanks to my supervisors, Dr. Alberto Suárez González and Dr. Gonzalo Martínez Muñoz. Without their unconditional support through all the steps of my PhD research, this thesis would have never been accomplished. I do not want to admire them just for their immense knowledge, but also for patience, motivation, and providing me an atmosphere where I could just focus on my research. Your advice on both research as well as on my career have been invaluable. I could not have imagined having better advisors for my Ph.D. study.

I would like to take this opportunity to thank Dr. José Ramón Dorronsoro Ibero, Dr. Pablo Varona and Dr. Daniel Hernández Lobato who have been very kind to help, whenever I approached them.

I gratefully acknowledge the members of my Ph.D. committee for their time. Many thanks go in particular to Dr. Luis Fernando Lago Fernández for his valuable feedback on a preliminary version of this thesis.

The thesis would not have come to a successful completion, without the help I received from the staff of the Department of Computer Sciences at Universidad Autónoma de Madrid. In particular, I would like to thank Amelia Martín for always being so helpful and friendly during all these years.

I would like also to express my gratitude to Dr. Ángela Fernández Pascual, Dr. Carlos María Alaíz Gudín and soon-to-be Doctor David Díaz Vico, for their friendship and academic advice. I thank also to all wonderful friends from the department: Eva, Jaime, Encarna, Santiago, Elena, Jocelyn and Hanna for being kind, supportive and caring friends.

This journey would not have been possible without the support of my family. My parents, my sister, my brothers and my beloved nephew, who provided unconditional love and care at every stage of my personal and academic life. Love you all!

Last but not least, I would like to say a heartfelt thank you to my little family my beloved husband, Vahid, and to my son, Aiden. Vahid, for his encouragement, attention, unconditional support and help in academic and non-academic issues; Aiden for being joyful and always cheering me up. There are no words to convey how much I love you both!

Contents

List of Figures

Dedicated to my lovely husband, Vahid
and our beloved son, Aiden

Chapter 1

Introduction

The availability of large amounts of data as a result of the widespread use of digital communications and interaction media (mobile communications, the Internet), obtained in systematic collection efforts (e.g., meteorological, environmental, and medical records) and from distributed sensor networks (e.g., the electricity distribution grid) poses new challenges in the extraction and processing of information from the raw inputs. Despite the large amount of information that can potentially be extracted from these data, learning by automatic induction from them can be rather challenging. One of the main difficulties that arises in learning is the poor quality of the data. In particular, contamination by noise is common in real-world situations [Frénay and Verleysen, 2014; Zhu and Wu, 2004a]. The presence of noise in the training data generally reduces the accuracy of the learned predictors. In general, noise is present in both the inputs (features, or attributes) and in the targets (e.g., class labels). From these two kinds of noise, typically, the latter has a more pronounced misleading effect than the former [Frénay and Verleysen, 2014]. The explanation of this observation is that, in general, noise in the target values causes a large distortion of the regularity patterns that are exploited for prediction. By contrast, noise in the input variables tends to simply blur these patterns. Provided that this blurring is small, it is possible to incorporate in the learning algorithms mechanisms to deal with such perturbations. In particular, feature selection or feature weighting techniques can be used to reduce the importance of noisy features in the learned predictive model. Notwithstanding, sizable levels of noise in the features can have as adverse an effect as target noise [Frénay and Verleysen, 2014; Zhu and Wu, 2004a].

Noise in the data can be handled either in the preprocessing phase (data cleansing) or during the induction process itself, by ensuring that the learning algorithms is robust [Frénay and Verleysen, 2014]. Ensemble methods can be used to build accurate models

that can be used for robust prediction and also to identify noisy instances [Brodley and Friedl, 1999; Freund, 2001; Khoshgoftaar et al., 2005; Zhu et al., 2003]. An ensemble consists of a collection of predictors whose outputs are combined in some manner (e.g. by averaging, in regression, or majority voting, in classification problems) to yield a final prediction. There is extensive empirical evidence showing that ensembles often outperform individual classifiers of the same type as the members of the ensemble. This is the case provided that the individual ensemble members are sufficiently accurate and tend to make errors on different examples [Breiman, 1996c; Oza and Tumer, 1999; Wolpert, 1992]. There are different reasons why ensembles can be more accurate than single learners [Dietterich, 1997]. The first reason is that the training data might not provide sufficient information for choosing a single best learner. For example, there may be a variety models that perform equally well on the training data. Instead of selecting one of them, combining the predictions of these learners can be a better choice. The second reason is that most learning algorithms involve some sort of search or optimization process. The algorithms used to this end could be imperfect. For example, even if a unique best hypothesis exists, it might be difficult to reach, because the algorithm becomes trapped in some sub-optimal solution (e.g. a local minimum). Ensembles can compensate for such deficiencies. The third reason is that, in most cases, the hypothesis space that is being searched does not contain the actual target function. In this case the combination of different hypothesis can effectively enlarge the space and provide a good approximation for the unknown predictor function. For example, the classification boundaries of standard decision trees (e.g. CART) are hyperplanes parallel to the coordinate axes. If the target classification boundary is not of this type, a single decision tree cannot provide a smooth approximation [Zhu and Wu, 2004a]. By contrast, a combination of decision trees can provide approximations to smooth boundaries of arbitrary shape.

According to these arguments, ensembles are expected to yield predictions that are both accurate and robust, even in the presence of noise in the training data. To have an effective ensemble, it should consist of complementary base predictors. A necessary condition for complementarity is diversity. Diversity can be achieved through direct manipulation of the training data (e.g. AdaBoost [Freund and Schapire, 1997]), variations in learner design (e.g. random forest [Breiman, 2001]), or by adding a penalty to the outputs (e.g. negative correlation learning [Liu and Yao, 1999]) to encourage diversity. There are some proposals that introduce diversity by means of artificial training examples [Melville and Mooney, 2005] or some other heuristic methods [Banfield et al., 2004; Partalas et al., 2008, 2012]. Switching the class labels of a randomly chosen fraction of examples is also an effective method to introduce diversity in the ensemble [Martínez-Muñoz and Suárez, 2005].

1.1 Thesis Objectives

In this thesis we have investigated the use of ensembles of classifiers to improve classification accuracy in the presence of class label noise. The main contributions are:

1. An analysis of two ensemble methods, bagging and random forests, when subsampling is used in the construction of their base learners. By using subsampling the base classifiers will be more diverse. As a consequence of this increased diversity, the classification margins generally decrease. However, in spite of common belief, these ensembles with small margins are more robust to noise than their counterparts which have larger margins. We have carried out an extensive empirical evaluation to assess the performance of these two methods in the presence of noise. Based on the results of this analysis, one concludes that using unpruned decision trees trained on bootstrap samples whose size is smaller than the training set size improves the resilience of bagging in the presence of class label noise. However, for random forest, since the base learners already present high variability, when standard bootstrap is used, subsampling is beneficial only in some datasets, except when the problem under consideration is contaminated by high noise levels.

2. A noise detection strategy is proposed based on building robust bootstrap ensembles, such as bagging and random forest, that use subsampling. Specifically, we propose a two stage method. In a first stage, the optimal sampling rate is estimated using out-of-bag data. Then, noisy instances are cleansed. A training instance is marked as noise when the fraction of incorrect predictions given by the ensemble members is above a threshold. The optimal value of this threshold is determined using a wrapper method. Finally, examples marked as noise are either removed (filtering) or relabeled (cleaning) and an ensemble is build anew.

3. Vote-boosting is a novel learning algorithm especially designed to build ensembles that are resilient to class-label noise. The central idea in the design is to use a measure of the uncertainty of the ensemble predictions to determine the type of emphasis to apply. Specifically, the weight of each training instance is determined in terms of the degree of disagreement among the predictions of the individual ensemble learners for that particular instance. In the implementation explored, the symmetric beta distribution is used as the emphasis function, in which the optimal shape parameter is determined by cross validation. Vote-boosting can be interpreted as a gradient descent algorithm in the hypothesis space of linear combinations of predictors.

4. Support Vector Machines are strong learners that are in general very accurate. Improving their accuracy using ensemble methods has proven to be an extremely

difficult task. One of the drawbacks of SVMs is that their generalization performance is very sensitive to the values of the hyperparameters of the method. These are typically determined through cross-validation, which is a time-consuming process. In this thesis, we propose to leverage on complementary randomization and optimization techniques to build ensembles of SVMs that are effective and can be build efficiently. Specifically, we take advantage of subsampling and the diversification of the optimized hyperparameter values. The computational cost of building such ensembles is much lower than training a single fully optimized SVM. Nevertheless, the accuracies of the ensemble and of the single SVM are comparable.

5. The type and number of base learners are important factors that determine the performance of heterogeneous ensemble. In this work we have considered ensembles composed of three different types of base learners: neural networks, SVMs, and random trees. By pooling classifiers of these types, built on different boostrap samples of the original training data, it is possible to build heterogeneous ensembles that interpolate between homogeneous ones. A near-optimal composition of such heterogeneous ensemble can be determined using out-of-bag data.

1.2 Thesis structure

This thesis is presented as an article compendium. It consists of three articles published in international peer-reviewed journals ([Sabzevari et al., 2015, 2018c,d]), a conference paper Sabzevari et al. [2018b], and an article available online [Sabzevari et al., 2018a]. The specific contribution of the author of this thesis to these publications has been: the design and discussion of the experiments and methods with the advisors; the implementation of the proposed methods and experiments; and, jointly with all other authors, the elaboration of the articles.

It is organized as follows:

- Chapter 2: This chapter provides an overview of ensemble learning and of the challenges that arise from the presence of noise in the training data.
- Chapter 3: In the investigation presented in this chapter subsampling is used to improve the robustness of bootstrap ensembles such as bagging or random forest. This strategy reduces the effect of noisy examples, which has a regularization effect on the learning process. In spite of exhibiting small classification margins, ensembles built in this manner are shown to be both robust and accurate, especially in the presence of class-label noise. This chapter corresponds to publication [Sabzevari et al., 2015]:

Maryam Sabzevari, Gonzalo Martínez-Muñoz, and Alberto Suárez. Small margin ensembles can be robust to class-label noise. Neurocomputing, 160 (Supplement C): 18 – 33, 2015. ISSN 0925-2312.

- Chapter 4: In this chapter, we propose a noise detection strategy based on the degree of disagreement among the predictions of the individual ensemble members. The work has been published as [Sabzevari et al., 2018c]:

  Maryam Sabzevari, Gonzalo Martínez-Muñoz, and Alberto Suárez. A two-stage ensemble method for the detection of class-label noise. Neurocomputing, 275:2374 – 2383, 2018. ISSN 0925-2312.

- Chapter 5: In this chapter, vote boosting is introduced and its performance evaluated. In this novel boosting method the degree of emphasis to the different training instances is determined on the basis of the tally of votes of the ensemble predictors. This type of emphasis does not depend on whether the predictions are erroneous or correct. As a consequence, the algorithm is not misled by the presence of class-label noise. This chapter corresponds to publication [Sabzevari et al., 2018d]:

  Maryam Sabzevari, Gonzalo Martínez-Muñoz, and Alberto Suárez. Vote-boosting ensembles. Pattern Recognition, 83:119 – 133, 2018. ISSN 0031-3203.

- Chapter 6: In this chapter a method for the construction of ensembles of SVMs is described. The method uses subsampling, randomization and optimization techniques for the diversification of the ensemble SVMs. This chapter has been presented as a conference paper in [Sabzevari et al., 2018b]:

  Maryam Sabzevari, Gonzalo Martínez-Muñoz, and Alberto Suárez. Randomization vs optimization in svm ensembles. In Věra Kůrková, Yannis Manolopoulos, Barbara Hammer, Lazaros Iliadis, and Ilias Maglogiannis, editors, Artificial Neural Networks and Machine Learning – ICANN 2018, pages 415–421, Cham, 2018b. Springer International Publishing. ISBN 978-3-030-01421-6.

- Chapter 7: In this chapter, we propose pooling the classifiers of homogeneous ensembles composed of base learners of different types to build a heterogeneous ensemble. The optimal fraction of learners of the different types in the heterogeneous ensemble is determined using out-of-bag error. This chapter corresponds to the following manuscript, which is available on-line [Sabzevari et al., 2018a]:

  Maryam Sabzevari, Gonzalo Martínez-Muñoz, and Alberto Suárez, Pooling homogeneous ensembles to build heterogeneous ensembles. CoRR, abs/1802.07877, 2018. URL http://arxiv.org/abs/1802.07877.

- Chapter 8: The final chapter of this thesis is devoted to a discussion of the results and conclusions of the investigation carried out. Potentially interesting research lines for further study are also proposed.

Chapter 2

Ensemble learning in noisy domains

Machine learning is a thriving field in computer science. The main goal of this research area is to design algorithms that can learn by automatic induction from data, without explicitly programming the solution. One is interested in identifying regularity patterns in the data that can be exploited to predict characteristics of data instances that have not been observed before (i.e. they *generalize* well). These algorithms are usually divided into two groups: Predictive, or supervised learning, and descriptive, or unsupervised learning [Murphy, 2012]. The goal of supervised learning is to construct a function that maps inputs to outputs from a set of known N_{train} input-output pairs

$$\mathcal{D}_{train} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N_{train}}, \tag{2.1}$$

where $\mathbf{x}_i \in \mathcal{X}$ is a D dimensional feature input vector, $y_i \in \mathcal{Y}$ is the output. One speaks of classification when the output takes values in a discrete set of class labels $\mathcal{Y} = \{c_1, \cdots, c_K\}$, where K is the number of different classes. Regression consists in predicting continuous ordered outputs, $\mathcal{Y} = \mathbb{R}$. In this thesis we will focus on classification tasks.

In the case of unsupervised learning, the inputs are available, but there is no information about the outputs. The main objective of unsupervised learning is to build a model of the underlying structure of the data [James et al., 2014]. Several tasks can be solved by unsupervised learning [Bishop, 2006]: clustering, in which one looks to find groups of similar inputs; probability density estimation; dimensionality reduction for data summarization and visualization, feature extraction, or further processing.

Machine learning has been used in applications from many different areas [Murphy, 2012]:

- Supervised learning :

  Classification : document classification [Afzal et al., 2015; Manevitz and Yousef, 2001], email spam filtering [Sculley and Wachman, 2007; Zhou et al., 2014] and handwriting recognition [Keysers et al., 2017; Plamondon and Srihari, 2000].

  Regression : prediction of stock market price [Kim and Han, 2000; Patel et al., 2015] and temperature predicting [Chen and Hwang, 2000; Jalil et al., 2017].

- Unsupervised learning: clustering [Faceli et al., 2007; Metsalu and Vilo, 2015], image inpainting [Bertalmio et al., 2003; Ružić and Pižurica, 2015] and market basket analysis [Kaur and Kang, 2016; Shaw et al., 2001].

The nature of a learning problem is determined by these aspects: data or sample space, hypothesis space and cost (loss) function [Vapnik, 1999]. Throughout this thesis we assume that the data instances are realizations of the random variables $(\mathbf{X}, Y)$ sampled independently from a fixed but unknown distribution S (i.e. they are iid). As hypothesis space $\mathcal{H}$, we consider the set of models such that each $h \in \mathcal{H}$ is a function that maps the set of input vectors $(\mathcal{X})$ into a set of discrete class labels $(\mathcal{Y})$, where hypothesis h partitions the input space $\mathcal{X}$ into K mutually exclusive regions. Each of these regions that are separated by decision boundaries corresponds to a different class. There are different loss functions that are used to quantify the cost associated with errors that result from using h to predict the class labels of the examples. A commonly used loss function in classification problems is the zero-one loss. However, the non-continuity and consequently non-convexity of the zero-one loss makes it impractical (intractable) for standard optimization frameworks. Hence, other convex alternatives such as the hinge-loss or the log-loss, have been proposed as surrogate loss functions that approximate the zero-one loss function.

In the context described, the objective of machine learning algorithms is to induce a hypothesis from the training examples that minimizes the risk associated to the selected loss function L. For known S, this can be expressed as

$$R(h) = \mathbf{E}[L(h(\mathbf{X}), Y)] = \int L(h(\mathbf{x}), y) dS(\mathbf{x}, y). \tag{2.2}$$

In classification problems, the generalization error is defined as the probability of misclassifying an instance $(\mathbf{x}, y)$ drawn from distribution S, is

$$Error(h) = Pr_{(\mathbf{X},Y)\sim S}[h(\mathbf{X}) \neq Y]. \tag{2.3}$$

The minimum generalization error, which is referred to as Bayes error, is achieved by optimal (Bayes) classifier. This classifier can be formulated as

$$h_S^*(\mathbf{x}) = \underset{Y}{\operatorname{argmax}}\, P(Y|\mathbf{X}). \tag{2.4}$$

where $P(Y|\mathbf{X})$ is given by Bayes rule $P(Y|\mathbf{X}) = \frac{P(\mathbf{X}|Y)P(Y)}{P(\mathbf{X})}$.

In most real-world problems the distribution S is unknown. Therefore, the risk given by equations 2.2, 2.3 and 2.4 cannot be calculated. Nevertheless an approximation of the actual generalization risk, the empirical risk, can be calculated by averaging the loss function on a set of labeled data as:

$$R_{emp} = \frac{1}{N}\sum_{j=1}^{N} L(h(\mathbf{x_j}) \neq y_j). \tag{2.5}$$

An estimate of the generalization error can also be given by computing the empirical error on a fresh (test) dataset, which is independent from the training set used to build the model. This test (or validation set) consists of N_{test} iid labeled pairs $\mathcal{D}_{test} = \{(\mathbf{x}_j, y_j)\}_{j=1}^{N_{test}}$, that are assumed to be drawn independently from the same distribution S as the training set. The empirical test error is defined as the average misclassification error of the instances in $\mathcal{D}_{test}$ [Mohri et al., 2012]

$$Error_{test}(h) = \frac{1}{N_{test}}\sum_{j=1}^{N_{test}} \mathbb{1}(h(\mathbf{x_j}) \neq y_j), \tag{2.6}$$

where $\mathbb{1}$ is the indicator function, which evaluates to 1 if the input statement is true and to 0 otherwise.

The generalization error can be decomposed in different ways [Breiman et al., 1984; Friedman, 1997; Kohavi and Wolpert, 1996]. The error of a learning algorithm $\mathcal{L}$ at a particular point $\mathbf{x}$ can be written as the squared bias plus the variance [Dietterich and Kong, 1995]:

$$Error(\mathcal{L}, \mathbf{x}, N_{train}) = bias(\mathcal{L}, \mathbf{x}, N_{train})^2 + variance(\mathcal{L}, \mathbf{x}, N_{train}), \tag{2.7}$$

where N_{train} is the size of the training data used to build S. The contribution of the bias to the error is given by the lack of expressive power of a learning algorithm with respect to the data at hand. Variance reflects the variability in the predictions due to fluctuations in the training data and, if it involves some randomization (e.g. the choice of initial weights of a neural network, or the random partition of the training data used

in cross-validation, the use of bootstrap sampling in ensemble methods, etc.), in the learning algorithm itself.

There is a natural bias-variance decomposition for regression problems: Let the labeled datasets $S_1, ..., S_T$, of size N_{train} be drawn independently from the original distribution. These sets are used to train the learning algorithm and to build hypotheses, $h_1, ..., h_T$. The average prediction at a specific point $\mathbf{x}$ is obtained by averaging over these hypotheses:

$$\overline{h}(\mathbf{x}) = \lim_{T \to \infty} \frac{1}{T} \sum_{i=1}^{T} h_i(\mathbf{x}). \tag{2.8}$$

The bias and variance for a particular instance $\mathbf{x}$ are defined as follows:

- Bias is the deviation of averaged hypothesis prediction from the actual label.

$$bias(\mathcal{L}, \mathbf{x}, N_{train}) = \overline{h}(\mathbf{x}) - f(\mathbf{x}), \tag{2.9}$$

  where $f(\mathbf{x})$ is the true label at point $\mathbf{x}$.

- Variance can be defined as the expected squared difference between a particular hypothesis i and the average hypothesis $\overline{h}(\mathbf{x})$:

$$Variance(\mathcal{L}, \mathbf{x}, N_{train}) = E\left[(h_i(\mathbf{x}) - \overline{h}(\mathbf{x}))^2\right] \tag{2.10}$$

There are other decompositions of the generalization error that consider also the Bayes error as a separate contribution [Breiman et al., 1984; Friedman, 1997; Kohavi and Wolpert, 1996]. However, as mentioned earlier in this thesis, the computation of Bayes error is generally not feasible since the underlying distribution of the data is often unknown. For classification, different definitions of bias and variance are possible [Domingos, 2000; Schiffner et al., 2012]. In all of them one attempts to break down the error into a part that is due to a systematic deviations of the predictions (bias) and a part that is the result of fluctuations (variance). Typically, predictors with high bias have low variance and vice versa.
The performance of machine learning models can be suboptimal in two manners: because of underfitting or due to overfitting. Underfitting occurs when the capacity of the applied learning model is limited. Consequently the model is not capable of capturing the complexity of the underlying data distribution. Typically, bias will be the dominant term in the error. Overfitting tends to appear when the model used are very flexible (low bias, high variance) and tend to memorize the training examples. Instead of regularity patterns, the model is misled by spurious regularities that are the result of fluctuations.

As a result, the predictor is not able to generalize and make accurate predictions on unseen examples.

In spite of the fact that different classifiers, which are based on a variety of theoretical principles and exhibit diverse characteristics, exist, it can be shown that their accuracy, when uniformly averaged over all possible classification problems, is equivalent. This is known as the *No Free Lunch Theorem* [Mitchell, 1980; Schaffer, 1994; Wolpert et al., 1995].

On a specific problem, a particular classifier can outperform others, because of its ability to capture specific patterns that are useful for prediction in that problem. In the context of classification problems, which are the focus of this thesis, some of the most popular learning methods are: neural networks (NN) [Hagan et al., 1996; Knerr et al., 1992], support vector machines (SVM) [Boser et al., 1992; Cortes and Vapnik, 1995a], k-nearest neighbours (k-NN) [Altman, 1992; Zhou et al., 2009], decision trees (DT) [Ding et al., 2002; Friedl and Brodley, 1997; Quinlan, 1986], ensemble learning [Breiman, 1996a, 2001; Dietterich, 2000a; Opitz and Maclin, 1999], etc. In most machine learning algorithms a single model is used to make predictions. In ensemble learning, the decisions of different models are combined in some manner to generate a final prediction. The goal of this thesis is the design of learning algorithms to build ensembles that are robust to the presence of class label noise in the training data. To this end we will explore the use of subsampling as a regularization mechanism in bootstrap ensembles, such as bagging or random forest. We also take advantage of randomization and optimization strategies to generate collections of diverse learners whose predictions are complementary.

2.1 Classification Margin

Margin maximization plays an important role in the analysis of many important machine learning methods such as AdaBoost [Rätsch and Warmuth, 2005] or Support Vector Machines (SVM) [Cortes and Vapnik, 1995b; Vapnik, 1998]. In general terms, the margin of an example can be defined as its distance to the decision boundary. In particular, for SVMs, the idea is to seek a separator hyperplane in an extended feature space that has an acceptable training accuracy while maximizing the distance with the training instances. In the context of ensemble learning, the margin is defined as the number of (weighted) votes for the correct class minus the number of (weighted) votes for the most voted incorrect class. In [Schapire et al., 1998b], it is shown that maximizing the margin plays an important role in AdaBoost success. In fact, AdaBoost iteratively maximizes the cumulative margin distribution for the training examples. Maximizing the margin can be an advantageous, in the sense small displacements of the examples

will not modify the predictions made by the model learned in this manner. It has been shown that increasing the margin on the training set can decrease the upper bound of the generalization error [Shawe-Taylor and Cristianini, 1998; Vapnik, 1998]. However, the direct optimization of the margin does not seem to be a universally effective method to build more accurate predictors Grove and Schuurmans [1998].

2.2 VC dimension and Structural Risk Minimization

Machine learning can be seen as a function estimation strategy using data. The quality of the learned functions can be affected by either underfitting or overfitting. These two obstacles can be overcome by choosing a model whose capacity is atuned to the complexity of the data available for learning. The capacity of a class of models (functions) can be quantified using the Vapnik-Chervonenkis (VC) dimension [Vapnik, 1991]. The VC dimension of a parametric family of functions is the maximum number of data instances that can be shattered by functions in that family. Shattering stands for separating instances irrespective of their class labels. A model with a large VC dimension is more complex and can shatter more points. For instance, k-nearest neighbors ($k = 1$) is a model with an infinite VC dimension. The simpler the hypothesis, the lower its VC dimension is. Complex models are prone to overfitting. Models that are too simple tend to underfit. The optimal complexity (i.e., VC dimension) of a model is a data-dependent. A possible option is to choose the simplest model that can classify the data sufficiently well. This learning bias is known as Occam's razor. In spite of being an important concept in statistical learning theory, calculating the VC dimension is usually not a trivial task (if at all possible) for most functions. In particular, estimating the VC dimension for non-linear models, such as neural networks, is rather difficult. For linear models, an upper bound on the VC dimension can be determined using the margin [Vapnik, 1999; Von Luxburg and Schölkopf, 2008]:

Theorem 2.1. *Let $\mathcal{H}$ be the space of linear classifiers in $\mathbb{R}^D$, when D is the dimensionality of the feature space, that separate the training data with margin at least ρ. Let r be the radius of the smallest ball in feature space that contains all instances. The VC dimension is bounded by*

$$VC(\mathcal{H}) \leq \min\{D, \frac{4r^2}{\rho^2}\} + 1. \tag{2.11}$$

This bound illustrates the inverse relation between the margin of the learning model and its VC dimension. In a general model, if an upper bound on the VC dimension can be given, it is possible to derive an upper bound of the generalization error using the following result:

Theorem 2.2. *Let V be the maximum VC dimension of a hypothesis $h \in \mathcal{H}$. Let $Error(h)$ denote the generalization error using the 0/1 loss. An unbiased estimate of this quantity computing $Error_{train}(h)$, the empirical error on a test set of size N_{train} and independent of the training data. For all $h \in \mathcal{H}$, the following bound holds with probability $1 - \mu$*

$$Error(h) \leq Error_{train}(h) + \sqrt{\frac{V(log(\frac{2N_{train}}{V}) + 1) - log(\frac{\mu}{4})}{N_{train}}}, \tag{2.12}$$

The second term on the right-hand side of this expression is called the VC confidence [Bartlett and Shawe-taylor, 1998]. Thus, small values VC dimension translates into high probabilities of having a lower generalization error.

Structural Risk Minimization (SRM) is a design principle for learning algorithms in which an optimal model is sought by striking a balance between the empirical error and the VC dimension In this approach one considers a sequence of nested classes of hypotheses whose VC dimensions are nondecreasing. In such a structure, each class of hypotheses in the sequence is more complex than the previous one. From these one selects the class with that minimized the generalization error upper bound given by Eq. (2.12). However, there are some limitations in computing an upper bound on the generalization error. As pointed out earlier, bounds on the VC dimension are not always easy to estimate, especially for non-linear models. Furthermore, the selection could involve a difficult-to-solve nonlinear optimization problem.

2.3 Ensemble Learning

Ensemble learning is a multi-learner paradigm in which several models are built and their predictions combined to yield a final decision. Each of the individuals in the ensemble is called a base learner. Ensemble methods can improve the accuracy of single learners by taking advantage of the complementarity among the individual predictions of their constituents. Ensemble algorithms aim at creating this complementary diversity among them[Ali, 1995; Banfield et al., 2004; Bishop, 2006; Breiman, 1996c; Brown et al., 2005; Dietterich, 2000a; Hansen and Salamon, 1990; Mohri et al., 2012; Windeatt, 2006].

For classification one of the simplest and most effective strategies for combining the base learners decisions is majority voting. In majority voting, the final prediction is the class label that is predicted most frequently by the individual ensemble classifiers. It can be shown that majority voting is an optimal combination strategy when individual

predictions are better than random guessing, independent, and $T \to \infty$, where T is the number of base learners [Hernández-Lobato et al., 2011; Lam and Suen, 1995, 1997].

In binary classification, when majority voting is used, the ensemble makes an error when at least $\lfloor T/2 \rfloor + 1$ of its base learners output an incorrect prediction. Let's assume that all individual learners have a probability of error p and that the errors are independent. Under these assumptions, the probability of error of the ensemble is the area under the binomial distribution where more than half of the base learners misclassify $\mathbf{x}$

$$Error_{test}(p, T) = \sum_{\lfloor t=T/2 \rfloor + 1}^{T} \binom{T}{t} p^t (1-p)^{T-t}. \tag{2.13}$$

As T approaches infinite, the ensemble error asymptotically tends to 0 if $p < \frac{1}{2}$. This result is known as the Condorcet Jury theorem [Vapnik, 1999]. Nevertheless, this theorem is not applicable in general as the individual errors of the classifiers are not completely independent in practice. A different scenario, is when we consider the case of classifying a single instance $\mathbf{x}$ by the ensemble. Given a training dataset and an instance to classify $\mathbf{x}$, then the outputs of the individual classifiers of the ensemble are independent and the probability of erring of the ensemble tend to a probability, $p_{\mathbf{x}}$, that depends on the difficulty of the instance Hernández-Lobato et al. [2009, 2011]; Hernández-Lobato et al. [2012]; Soto et al. [2016].

A variety reasons explain why ensemble learning works in practice [Dietterich, 2000a]:

- Statistical: A learning algorithm can be transformed to an optimization problem whose goal is to find the best $h \in \mathcal{H}$ in space of hypotheses $\mathcal{H}$. Consider the scenario in which the available data are not sufficient to identify the proper hypothesis. In that case, there might be several hypotheses with comparable accuracy. By averaging over all these hypotheses we limit the effect of selecting a suboptimal classifier.

- Computational: A common scenario in optimization problems is that the objective function has several local minima. In such cases, even when there are enough data, the learning algorithm may converge to one of these suboptimal local minima for computational reasons. For example finding the neural network or the decision tree that minimize a general cost-complexity cost function is typically an NP-hard problem [Mohri et al., 2012]. Combining the predictions of such suboptimal learners, which are found by local searches with different starting points, can be beneficial to reduce the effect of getting stuck in a single local minimum.

- Representational: The representational capacity of a hypothesis space can be extended beyond the search space by averaging over several hypotheses. Thus an ensemble can learn patterns outside $\mathcal{H}$.

A homogeneous ensemble is composed of predictors of the same type (e.g., neural networks, decision trees, or SVM's). Different diversification strategies can be used to build the individual predictors. In particular, different versions of the training set for the base learners can be used, as in bagging (bootstrap sampling of training data), class-switching (noise injection in the class labels of the training instances), or in the random subspace method (subsampling the feature space). Another possibility is to use different settings of the learning algorithm: use different values of the hyperparameters, different architectures, etc. If the base learning algorithm involves the optimization of a cost function that can have different local minima, different seeds for the model parameters (e.g. the synaptic weights of a neural network) can be used. In sequential homogeneous ensemble methods, such as boosting, the learning algorithm can be formulated as an optimization problem so that each additional classifier is built to improve the prediction of the current ensemble.

Heterogeneous ensembles are a collection of base learners of different types. (e.g. neural networks, decision trees, and SVM's). Heterogeneous ensembles have an intrinsic diversification mechanism because base classifiers of different types are likely to have different biases. However, it is important to determine which combinations of base classifiers are more effective. In addition to the predictor types, a second important choice in heterogeneous ensembles is to find the optimal proportion of each type to be combined into the ensemble [Partalas et al., 2010]. In general, two strategies have been proposed to build heterogeneous ensembles. In the first strategy a fixed number of different models are combined [de Oliveira et al., 2013; Nanni et al., 2015]. A second strategy is to build a group of classifiers with different parameterizations and then select the best subset to include in the final ensemble [Caruana et al., 2004; Haque et al., 2016; Partalas et al., 2010].

In what follows we provide a brief overview of the different types of ensembles that are considered in this thesis. Namely, bagging, random forests and related methods, class-switching, and boosting ensembles.

2.3.1 Bagging

Bagging [Breiman, 1996c] is an ensemble method in which the base learners are built using different bootstrap samples of the original training data. Bootstrap sampling consists in drawing randomly, with replacement from the original training set (resampling

with replacement) N_{train} instances, where N_{train} is the size of the training data. In each draw, a training instance is selected with probability $1/N_{train}$. After N_{train} draws the probability that a particular instance in the original training dat is selected is

$$1 - (1 - \frac{1}{N_{train}})^{N_{train}} \approx 1 - e^{-1} = 0.632. \tag{2.14}$$

Hence, on average approximately 63.2% of the instances in each bootstrap sample are unique. The rest are repeated instances. For classification, the final ensemble decision is given by majority voting. The pseudo-code of bagging is shown in algorithm (1). The diversity among base learners in bagging is achieved by using different bootstrap samples of the original training set, which are not disjoint. As discussed earlier in this chapter, the error of a learning algorithm error can be decomposed into bias and variance terms. In bagging, one expects that the accuracy improvement comes from the variance reduction entailed by pooling the outputs of the different base learners [Bauer and Kohavi, 1999]. This variance reduction should be more pronounced for unstable learners (e.g. decision trees). Due to their low variability bagging is not expected to be effective when the base learners are stable [Breiman, 1996c]. For instance, bagging is not very efficient when applied to stable learners such as decision stumps (which are trees with depth 1), due to the similarity of the linear decisions given by the individual base learners [Zhu et al., 2008].

Algorithm 1: Bagging algorithm[Bauer and Kohavi, 1999]

Input: $\mathcal{D}_{train}=\{(\mathbf{x}_i, y_i)\}_{i=1}^{N_{train}}$ % Training set
T % Ensemble size
$\mathcal{L}$ % Base learning algorithm
for $t \leftarrow 1$ ***to*** T **do**
  $D_t \leftarrow Bootstrap(\mathcal{D}_{train})$
  $h_t(\cdot) \leftarrow \mathcal{L}(D_t)$
Output: $H(\cdot) = \arg\max_y \sum_{t:h_t(\cdot)=y} 1$
% Majority vote

An alternative strategy is to use sampling without replacement, in which instances in each sample are unique draws from the original training set. Subbagging ensembles are built using this method [Bühlmann and Yu, 2002]. A common choice is to draw $\frac{N_{train}}{2}$ unique examples from the original training set in each sample [Friedman and Hall, 2007]. An interesting observation is that, in terms of second order statistics, subbagging with samples of size $\frac{N_{train}}{2}$ is equivalent to standard bagging [Buja and Stuetzle, 2006; Friedman and Hall, 2007; Martínez-Muñoz and Suárez, 2010]. An advantage of subbaging over the equivalent bagging method is the reduction of the computational costs building the ensemble [Martínez-Muñoz and Suárez, 2010]. Of course, one is not limited to using

samples of size N_{train} in bagging or $N_{train}/2$ in subbaging. Other choices are possible. Different sampling ratios in bagging and subbagging can be used to the accuracy of the ensemble [Martínez-Muñoz and Suárez, 2010]. Furthermore, in Chapters 3 and 4 of this thesis, we show that subsampling is an effective mechanism to improve also the robustness of ensemble learning in the presence of class-label noise [Sabzevari et al., 2014].

Next, we discuss the reason why averaging the predictions of base learners has the potential to improve the generalization capability of the ensembles. The presentation is made in the context of regression. A similar analysis can be made for classification [Breiman, 1996c].

Let $\mathbf{x}$ be an example with unknown output variable. The ensemble expected prediction for this instance is

$$H(\mathbf{x}) = \mathbb{E}_{\mathcal{D}_{train}}[h(\mathbf{x})], \tag{2.15}$$

where D_{train} refers to an average over all possible realizations of the training set. Let's assume that y is the actual value of the continuous target for the input vector $\mathbf{x}$. The mean square prediction error of hypothesis $h(\mathbf{x})$ is

$$\mathbb{E}_{\mathcal{D}_{train}}[(h(\mathbf{x}) - y)^2] = y^2 - 2y\mathbb{E}_{\mathcal{D}_{train}}[h(\mathbf{x})] + \mathbb{E}_{\mathcal{D}_{train}}[h^2(\mathbf{x})]. \tag{2.16}$$

Using the fact that $\mathbb{E}_{\mathcal{D}_{train}}[h^2(\mathbf{x})] \geq (\mathbb{E}_{\mathcal{D}_{train}}[h(\mathbf{x})])^2$, we can write

$$\mathbb{E}_{\mathcal{D}_{train}}[(h(\mathbf{x}) - y)^2] \geq y^2 - 2y\mathbb{E}_{\mathcal{D}_{train}}[h(\mathbf{x})] + (\mathbb{E}_{\mathcal{D}_{train}}[h(\mathbf{x})])^2. \tag{2.17}$$

Using the aggregated predictor definition in (2.15),

$$\mathbb{E}_{\mathcal{D}_{train}}[(h(\mathbf{x}) - y)^2] \geq y^2 - 2yH_A(\mathbf{x}) + [H_A(\mathbf{x})]^2 = (H_A(\mathbf{x}) - y)^2. \tag{2.18}$$

This means that the average prediction has smaller or equal mean squared error than the expected squared error of the individual predictions. The improvement in performance is given by the difference between $\mathbb{E}_{\mathcal{D}_{train}}[h(\mathbf{x})^2]$ and $(\mathbb{E}_{\mathcal{D}_{train}}[h(\mathbf{x})])^2$. Note that when the hypotheses $h(x)$ produce the same outputs, irrespective of the realization of $\mathcal{D}_{train}$, the two expectations converge to the same value. Therefore, unless there is some diversity among the hypotheses learnt from different realizations of $\mathcal{D}_{train}$, aggregation will not leads to improvements with respect to an individual base learner [Breiman, 1996c]. In bagging the average is not over different realizations of the training data, but over different bootstrap samples from a single training set

$$H_B(\mathbf{x}) = \mathbb{E}_{\mathcal{D}_t(\mathcal{D}_{train})}[h(\mathbf{x})] \simeq \frac{1}{T}\sum_{t=1}^{T} h_t(\mathbf{x}) \tag{2.19}$$

where $\mathcal{D}_t(\mathcal{D}_{train})$ is the bootstrap sample drawn from the original training set. This bootstrap estimate is generally not as accurate as $H(\mathbf{x})$. Note that, the base learners trained with bootstrap samples are generally less accurate than their counterparts trained with the original training data, due to using fewer unique training examples. Notwithstanding, the aggregation process usually tends to lower the prediction error by variance reduction.

2.3.2 Random forest, extremely randomized trees and rotation forest

Random forest (RF) [Breiman, 2001] is an ensemble method that combines decision trees built using two different randomization strategies. First, each tree is trained on a bootstrap sample drawn from the original training data, as in bagging. Second, at each internal node of the decision trees, a random subset of the features is considered to find the best split. The size of the random subset of features is a parameter that is typically set to $\sqrt{D}$, where D is the number of input features. Using a subset of the original features to determine the optimal splits increases the chance of having diversity. As a result of this additional diversification mechanism the ensemble can be more effective. Furthermore, since the search space for each split is smaller, and no post-pruning is made, the individual trees are built faster than in bagging. In practice, this method is an effective and robust algorithm in both classification and regression problems [Breiman, 2001; Caruana and Niculescu-Mizil, 2006; Caruana et al., 2008; Gislason et al., 2006]. In fact, this method is known as one of the best off-the-shelf classification methods for non-structured data [Fernández-Delgado et al., 2014a]. The pseudo-code of RF is shown in algorithm (2).

Algorithm 2: Random Forest algorithm[Breiman, 2001]

Input: $\mathcal{D}_{train}=\{(\mathbf{x}_i, y_i)\}_{i=1}^{N_{train}}$ % Training set
T % Ensemble size
d % number of selected features in each split

1 **for** $t \leftarrow 1$ ***to*** T **do**
2 $\quad D_t \leftarrow Bootstrap(\mathcal{D}_{train})$
3 $\quad h_t(\cdot)$ = Random Tree (d, D_t)

Output: $H(\cdot) = \arg\max_{y} \sum_{t:h_t(\cdot)=y} 1$ %Majority voting

Random forests have been studied extensively in the literature [Genuer et al., 2017; Geurts et al., 2006; Guha et al., 2016; Lakshminarayanan et al., 2014; Robnik-Šikonja, 2004]. Several variants of the method have been proposed. An interesting variant is Extremely randomized trees [Geurts et al., 2006]. In this method, in each node of the tree the split attribute and the split threshold are chosen at random. This extra

randomization can increase the ensemble performance by increasing the diversity. It can be shown that in this method the bias increases and the variance decreases [Geurts et al., 2006]. To avoid a further increase of the bias, the complete original training data is used to build each tree. The randomization level is controlled by an additional hyperparameter, whose value needs to be carefully determined. In an extreme scenario, in each node a single attribute is drawn at random and the the splitting is also made at random. In this case the tree structure is independent of the given targets. Therefore, there is a significant increase of the bias that cannot be compensated by the reduction of the variance. If the randomization level is properly adjusted, the variance can almost be reduced to almost zero with a negligible increment of the bias [Geurts et al., 2006].

Another ensemble method that belongs to the family of random forest is rotation forest [Rodríguez et al., 2006]. In this method diversity is generated by the construction of additional features by performing rotations. Specifically, to train each particular base learner, the set of D features are divided into K subsets. For each subset, principal component analysis (PCA) is performed on a bootstrap sample containing a fraction (typically 75%) of the training instances in that subset. After calculating the projection matrix of PCA, the whole training set with the new extracted features from all subsets is used to train each base learner. The process continues by building the new base learners using the feature sets extracted from the new K splitting of subsets. A common choice for the base learners are decision trees, because their sensitivity to rotation of axes. Improvements over AdaBoost, random forest and bagging are reported in [Rodríguez et al., 2006].

2.3.3 Class-switching ensembles

Noise injection in the labels of the training instances is another technique that can be used to build collections of diverse base learners. In this method, which was originally proposed by Breiman in [Breiman, 2000], each base learner is built on a version of the training set in which the class labels of $p\%$ randomly selected instances are changed also at random. In [Breiman, 2000] the class labels the labels were switched in such a way that the proportion of classes did not change. Later studies show that a fully randomized class-swithcing strategy, in which the proportion of classes in the original problem may change for unbalanced problems, is actually more effective [Martínez-Muñoz and Suárez, 2005].

2.3.4 Boosting

Boosting is an appealing idea to build ensembles of classifiers as it has a strong theoretical support [James et al., 2014]. Boosting originally was proposed as an answer to a question raised in Probably Approximately Correct (PAC) framework in the late 80's. The question was whether weak learners, whose accuracy is only slightly better than random guessing, can be combined in a prediction system that is strong and makes accurate predictions.

The most common boosting algorithm, AdaBoost, was proposed in 1997 [Freund and Schapire, 1997]. In AdaBoost an ensemble is grown by incorporating classifiers that progressively focus on instances that are misclassified by the previous classifiers in the sequence. The pseudo-code of AdaBoost is shown in Algorithm (3).

Algorithm 3: AdaBoost algorithm [Freund and Schapire, 1997]

Input: $\mathcal{D}_{train}=\{(\mathbf{x}_i, y_i)\}_{i=1}^{N_{train}}$ % Training set
T % Ensemble size
$\mathcal{L}$ % Base learning algorithm

$w_1(\mathbf{x}_i) \leftarrow \frac{1}{N_{train}}$ for $i = 1, ..., N_{train}$ % Initialize the vector of weights
1 **for** $t \leftarrow 1$ ***to*** T **do**
2 $D_t \leftarrow Bootstrap(\mathcal{D}_{train}, \mathbf{w}_t)$% Bootstrap sample from $\mathcal{D}_{train}$ with weights $\mathbf{w}_t$
3 $h_t(\cdot) \leftarrow \mathcal{L}(D_t)$ $\epsilon_t = Pr_{\mathbf{w}_t}[h_t(\mathbf{x}_i) \neq y_i]$
4 **if** $\epsilon_t > \frac{1}{2}$ **then**
5 $T = t - 1$ and exit loop
6 $\alpha_t = 1/2 \ln(\frac{1-\epsilon_t}{\epsilon_t})$ % The weight of base learner t
7 **for** $i \leftarrow 1$ ***to*** N_{train} **do**
8 $w_{t+1}(i) = \frac{w_t(i) exp(-\alpha_t y_i h_t(\mathbf{x}_i))}{Z_t}$
9 % $Z_t = \sum_{j=1}^{N_{train}} w_t(j) e^{-\alpha_t y_j h_t(\mathbf{x}_j)}$ is a normalization factor

Output: $H(\cdot) = sign(\sum_{t=1}^{T} \alpha_t h_t(\cdot))$

AdaBoost works by iteratively assigning weights to the training instances. First, the weights are initialized to be uniform. A base learner is trained with either a sample drawn from the original training data using the weight distribution (weighted resampling strategy) or using directly the weighted examples (reweighting strategy). If the weighted error of the base learner is greater than 0.5 (line 5), the algorithm terminates (line 6), otherwise the coefficient of the base learner for the final ensemble decision is calculated using this weighted error. The higher the error of base learner the smaller its coefficient (importance) in the ensemble prediction. Weights of the training examples are updated using an exponential function multiplied by the previous weight, in such a way that

the weights of the misclassified instances are increased and the weights of the correctly classified instances are decreased. The ensemble prediction is computed by the weighted majority voting [Freund and Schapire, 1997].

One of the main drawbacks of AdaBoost, which has been addressed by many studies, is sensitivity to noise in the class-labels [Cao et al., 2012; Cheamanunkul et al., 2014; Friedman et al., 2000; Gómez-Verdejo et al., 2008; Jiang, 2001; Loureiro et al., 2004; Shivaswamy and Jebara, 2011]. The reason for this sensitivity is that boosting tends to assign higher weights to instances that are difficult to classify. However, some of these instances could be outliers or simply be incorrectly labeled. This undue emphasis on incorrectly labeled instances can bring about a strong distortion of the original classification problem which has a deleterious effect on learning [Bauer and Kohavi, 1999; Quinlan, 1996].

AdaBoost has been investigated from different points of view and applied in many practical studies [Hu et al., 2008; Khammari et al., 2005; Lee et al., 2011; Lv and Nevatia, 2006; Wang and Wang, 2008]. One of the main reasons of the broad use of this method is its strong theoretical basis, the simplicity of its implementation, and the excellent accuracy in many classification problems. In [Bauer and Kohavi, 1999], the reasons for the superior performance of AdaBoost with respect to bagging are investigated. An interesting observation of that analysis is that, when trees are used as base classifiers, AdaBoost reduces both variance and bias, whereas in bagging only variance is reduced. In addition, they observed that variance reduction in AdaBoost is larger than than bagging. This observation is also supported by [Breiman, 1998].

AdaBoost has also been analyzed as a margin maximizing learning method [Schapire et al., 1998b]. It has been shown that in AdaBoost the generalization accuracy continues to improve with size of the ensemble even after the training error saturates. This behavior is connected to the observation that AdaBoost continues to increase the margin of the training examples with the number of classifiers, even after all the training instances are correctly classified [Schapire et al., 1998b]. However, the effectiveness of margin maximization in minimizing generalization error is unclear. Breiman proposed arc-gv, an algorithm that achieves a larger minimum margin than AdaBoost, but has a lower generalization accuracy [Breiman, 1999]. After that, Reyzin and Schapire defended the importance of increasing the margin in AdaBoost by emphasizing that AdaBoost maximizes the margin distribution rather than the minimum margin [Reyzin and Schapire, 2006]. In addition, they noticed that Breiman had used more complex base learners than the ones used in their empirical analysis. This is an important issue since they present an upper bound on generalization error that depends not only on the margin, but also

on the complexity of the base classifiers. Increasing the complexity of the base learners increases the generalization error, because of the increased likelihood of overfitting [Reyzin and Schapire, 2006].

In the context of ensemble learning, the process of incorporating a base classifier in the combination can be seen as an optimization step to reduce a cost functional. Two pioneering studies showed that AdaBoost can be can be viewed as a gradient descent method to minimize a cost functional in the functional space of linear combinations of hypotheses [Friedman, 2000; Mason et al., 1999a]. Friedman *et al.* by taking this observation as a starting point Mason et al. [1999a] proposed a general boosting algorithm called *anyboost*, in which different loss functions can be used. Specifically, in anyboost, one seeks for a linear combination of base learners in an inner product functional space to minimize some cost functional.

Consider a binary classification problem. The base learners are selected from a space of functions $\mathcal{F}$, where $f_\tau \in \mathcal{F}$ and $f_\tau : \mathcal{X} \to \{-1, 1\}$. The ensemble output is given by the sign of the weighted aggregation of the outputs of the base learners

$$H_T(\mathbf{x}) = sign[F_T(\mathbf{x})], \tag{2.20}$$

where

$$F_T(\mathbf{x}) = \sum_{t=1}^{T} \alpha_t f_t(\mathbf{x}), \quad F_T : \mathcal{X} \to \mathbb{R}. \tag{2.21}$$

In this expression α_t determines the weight (importance) of the classifier t in the ensemble aggregation. Note that $F_T \in lin(\mathcal{F})$, where $lin(\mathcal{F})$ is the set of linear combinations of functions in $\mathcal{F}$.

Define the cost functional $C[F] : lin(\mathcal{F}) \to \mathbb{R}^+$ of the form

$$C[F] = \int_{\mathcal{X},\mathcal{Y}} c(yF(\mathbf{x}))dP(\mathbf{x}, y) \tag{2.22}$$

where c is a misclassification cost function.

Let $\mathcal{D}_{train} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N_{train}}$ be a set of N_{train} label examples. Each training example is composed of a vector of attributes $\mathbf{x}_i \in \mathcal{X}$ and its corresponding class label $y_i \in \{-1, 1\}$. An empirical estimate of the cost functional on this training set

$$\hat{C}[F] = \frac{1}{N_{train}} \sum_{i=1}^{N_{train}} c(y_i F(\mathbf{x}_i)), \tag{2.23}$$

For a given $F_T \sum_{t=1}^{T} \alpha_t f_t(\mathbf{x})$, the objective is to find the values of f $\alpha_{T+1} \in \mathbb{R}$ and $f_{T+1} \in \mathcal{F}$ that minimize $\hat{C}[F_{T+1}]$, where

$$F_{T+1}(\mathbf{x}) = F_T(\mathbf{x}) + \alpha_{T+1} f_{T+1}(\mathbf{x}). \tag{2.24}$$

Assuming that $F_{T+1} - F_T$ is small, to first order in this difference

$$\begin{aligned} \hat{C}[F_{T+1}] &= \hat{C}[F_T + \alpha_{T+1} f_{T+1}] = \frac{1}{N_{train}} \sum_{i=1}^{N_{train}} c\left(y_i \left(F_T(\mathbf{x}_i) + \alpha_{T+1} f_{T+1}(\mathbf{x}_i)\right)\right), \\ &\approx \frac{1}{N_{train}} \sum_{i=1}^{N_{train}} \left[c\left(y_i \left(F_T(\mathbf{x}_i)\right)\right) + \alpha_{T+1} f_{T+1}(\mathbf{x}_i) c'\left(y_i \left(F_T(\mathbf{x}_i)\right)\right)\right]. \end{aligned} \tag{2.25}$$

Assuming that $c(z)$ is a monotonically decreasing function of z, which is consistent with its interpretation as a cost function, and that $\alpha_{T+1} > 0$ is fixed, the value of f_{T+1} that minimizes the value of the cost functional is the one that miminizes

$$\begin{aligned} \frac{N_{train}}{\alpha_{T+1}} \delta C[F_T] &= \frac{N_{train}}{\alpha_{T+1}} \left(C[F_{T+1}] - C[F_T]\right) = \sum_{i=1}^{N_{train}} c'(y_i F_T(\mathbf{x}_i)) y_i f(\mathbf{x}_i) \\ &= \sum_{i:y_i = f(\mathbf{x}_i)} c'(y_i F_T(\mathbf{x}_i)) - \sum_{i:y_i \neq f(\mathbf{x}_i)} c'(y_i F_T(\mathbf{x}_i)) \\ &= \sum_{i=1}^{N_{train}} c'(y_i F_T(\mathbf{x}_i)) - 2 \sum_{i:y_i \neq f(\mathbf{x}_i)} c'(y_i F_T(\mathbf{x}_i)) = \\ &= \sum_{i=1}^{N_{train}} c'(y_i F_T(\mathbf{x}_i)) \left(1 - 2 \sum_{i:y_i \neq f(\mathbf{x}_i)} \frac{c'(y_i F_T(\mathbf{x}_i))}{\sum_{j=1}^{N_{train}} c'(y_j F_T(\mathbf{x}_j))}\right) \\ &= -2 \sum_{i=1}^{N_{train}} c'(y_i F_T(\mathbf{x}_i)) \left(\sum_{i:y_i \neq f(\mathbf{x}_i)} w_i^{[T+1]} - \frac{1}{2}\right), \end{aligned} \tag{2.26}$$

where

$$w_i^{[T+1]} = \frac{c'(y_i F_T(\mathbf{x}_i))}{\sum_{j=1}^{N_{train}} c'(y_j F_T(\mathbf{x}_j))} \geq 0, \quad i = 1, 2, \ldots, N_{train}. \tag{2.27}$$

Since c is a monotonically decreasing, $-c'$ is non-negative, and $\{w_i^{[T+1]}\}_{i=1}^{N_{train}}$ can be regarded as the weights of the instances. Therefore, minimizing (2.24) for F_T and α_{T+1} fixed is equivalent to minimizing the weighted error

$$\epsilon_{T+1} = \sum_{i:y_i \neq f(\mathbf{x}_i)} w_i^{[T+1]}. \tag{2.28}$$

The stopping criterion is

$$\delta C[F_T] \geq 0 \quad \Longrightarrow \quad \sum_{i:y_i \neq f_{T+1}(\mathbf{x}_i)} w_i^{[T+1]} \geq \frac{1}{2}. \tag{2.29}$$

Assume that, given F_T, we have selected the optimal f_{T+1}. The value of α_{T+1} is determined by minimizing

$$C[F_T + \alpha f_{T+1}] = \frac{1}{N_{train}} \sum_{i=1}^{N_{train}} c(y_i(F_T(\mathbf{x}_i) + \alpha f_{T+1}(\mathbf{x}_i))), \tag{2.30}$$

with respect to α. Taking derivative of this expression with respect to α and setting the value of this derivative to 0 at $\alpha = \alpha_{T+1}$, one gets

$$\frac{1}{N_{train}} \sum_{i=1}^{N_{train}} c'(y_i(F_T(\mathbf{x}_i) + \alpha_{T+1} f_{T+1}(\mathbf{x}_i)))y_i f_{T+1}(\mathbf{x}_i) = 0. \tag{2.31}$$

Therefore, the solution α_{T+1} of this equation satisfies the relation

$$\sum_{i:f_{T+1}(x_i)=y_i} c'(y_i F_T(x_i) + \alpha_{T+1}) = \sum_{i:f_{T+1}(x_i)\neq y_i} c'(y_i F_T(x_i) + \alpha_{T+1}). \tag{2.32}$$

In the case of AdaBoost the misclassification function is

$$c(z) = e^{-z}, \qquad c'(z) = -e^{-z}. \tag{2.33}$$

Therefore, the instance weights are

$$w_i^{[T+1]} = \frac{e^{-y_i F_T(\mathbf{x}_i)}}{\sum_{j=1}^{N_{train}} e^{-y_j F_T(\mathbf{x}_j)}} = \frac{e^{-y_i \sum_{t=1}^{T} \alpha_t f_t(\mathbf{x}_i)}}{\sum_{j=1}^{N_{train}} e^{-y_j \sum_{t=1}^{T} \alpha_t f_t(\mathbf{x}_j)}} \quad i = 1, \cdots, N_{train} \tag{2.34}$$

The solution of Eq. (2.32) is

$$e^{2\alpha_{T+1}} = \frac{\sum_{i:f(\mathbf{x}_i)=y_i} e^{-y_i F_T(\mathbf{x}_i)}}{\sum_{i:f(\mathbf{x}_i)\neq y_i} e^{-y_i F_T(\mathbf{x}_i)}} = \frac{\sum_{i:f(\mathbf{x}_i)=y_i} w_i^{[T+1]}}{\sum_{i:f(\mathbf{x}_i)\neq y_i} w_i^{[T+1]}} = \frac{1 - \epsilon_{T+1}}{\epsilon_{T+1}}. \tag{2.35}$$

Solving for α_{T+1}, one gets

$$\alpha_{T+1} = \frac{1}{2} log \frac{1 - \epsilon_{T+1}}{\epsilon_{T+1}}. \tag{2.36}$$

Now α_{T+1} can be substituted in the expression of the weights update expression

$$w_i^{[T+1]} = \frac{e^{-y_i F_T(\mathbf{x}_i)}}{\sum_{j=1}^{N_{train}} e^{-y_j F_T(\mathbf{x}_j)}} = \frac{e^{-y_i \sum_{t=1}^{T} \alpha_t f_t(\mathbf{x}_i)}}{\sum_{j=1}^{N_{train}} e^{-y_j \sum_{t=1}^{T} \alpha_t f_t(\mathbf{x}_j)}} = \frac{1}{Z^{[T+1]}} w_i^{[T]} e^{-y_i \alpha_{T+1} f_T(\mathbf{x}_i)}, \quad (2.37)$$

where

$$\begin{aligned} Z^{[T+1]} = \sum_{i=1}^{N_{train}} w_i^{[T]} e^{-y_i \alpha_T f_T(\mathbf{x}_i)} &= \sum_{i:f(\mathbf{x}_i)=y_i} w_i^{[T]} e^{-\alpha_T} + \sum_{i:f(\mathbf{x}_i)\neq y_i} w_i^{[T]} e^{\alpha_T} \\ &= (1-\epsilon_T)\sqrt{\frac{\epsilon_T}{1-\epsilon_T}} + \epsilon_T \sqrt{\frac{1-\epsilon_T}{\epsilon_T}} = 2\sqrt{\epsilon_T(1-\epsilon_T)}. \end{aligned} \quad (2.38)$$

2.3.5 Gradient boosting

Another effective boosting method is gradient boosting [Friedman et al., 2000]. In this method, a differentiable loss function (e.g. the squared error) is minimized using an iterative procedure. Gradient boosting was originally developed for regression. Nevertheless, it is possible to extend it so that it can be applied to classification. As in boosting, the output of the ensemble is a weighted combination of the outputs of the individual regressors. At each iteration a new base regressor is fitted to the pseudo-residuals on the training instances of the predictions by the current ensemble. Pseudo-residuals for each training point are calculated by applying the partial derivative of the loss function with respect to the hypothesis, that consists in a linear additive model. At each iteration, a new classifier is built to fit the pseudo-residuals. The weight of the new base classifier is estimated using a line search to minimize the loss function. The new base classifier is incorporated into the ensemble and the pseudo-residuals are computed for the next iteration. Intuitively, gradient boosting attempts at each iteration to adjust the misestimations made by the current ensemble. The main drawback of gradient boosting is its tendency to overfit. Beyond a point, the incorporation of additional base classifiers greedily to minimize the error made on the training data can result in a degradation of the ensemble's generalization ability. Regularization strategies can be used to limit overfitting. For instance, shrinkage can be employed to reduce the contribution of each base classifier. Additionally, one can use simpler base learners, or stochastic gradient boosting (subsampling) [Friedman, 2002]. Extreme gradient boosting (XGBoost) [Chen et al., 2015] is an implementation of gradient boosting in which overfitting is addressed by including a complexity term in the cost function that is minimized. In an ensemble of trees, this term translates into a criterion that controls the number of tree nodes.

2.4 Learning in the presence of noise

In recent times massive amounts of data, recorded by many newly developed strategies and devices are at our disposal for analysis. However, in many of the cases, poor data quality due to the presence of noise, makes learning from these datasets rather challenging [Frénay and Verleysen, 2014; Zhu and Wu, 2004a]. Noise can be inherent to the process of data collection, such a noise arising from a recording camera in image or video data, or it can be the effect of a deliberate contamination. In the former case the distortion is usually referred to as random noise. The latter is characterized as adversarial noise. In this thesis we address the problem of supervised learning in the presence of random noise in the class labels. Noise can also be present in attributes. In general, noise in the class labels has a large deleterious effect in learning. Attribute noise is typically less noxious, except when a large number of features are contaminated with noise or high levels of noise cause a sizable distortion in attribute space. [Frénay and Verleysen, 2014; Sáez et al., 2014; Zhu and Wu, 2004a].

In [Frénay and Verleysen, 2014], Frénay *et al.* three different cases are described in terms of the statistical properties of the noise in the class labels

- Noisy Completely at Random Model (NCAR): The probability of having a modified class label is independent of feature values and of the true class label. In this model there is no class label or feature set more prone to noise than others.
- Noisy at Random (NAR): Label errors depend only on the true label. This model considers that some classes are more affected by noise than others.
- Noisy Not at Random (NNAR): The probability of a labeling error depends on the true class label and on the values of the features. In this model some regions in the feature space regions are more prone to being noisy than others (e.g. sparse regions or regions near to boundaries).

In general terms, there are two types of strategies to deal with noisy data. The effects of noise can be mitigated by *cleansing* the data before the actual learning process is carried out. Alternatively, robust learning methods that can deal with noise can be used for induction from the contaminated data [Frénay and Verleysen, 2014]. Data cleansing is a preprocessing step in which noise detection approaches are used to identify examples that are suspected to be noisy. Then, the detected examples can be either removed from the training set (*filtering*) or cleaned. Cleaning consists in correcting the class labels of the instances identified as noisy. Different approaches can be used for the detection of noise: density estimation, clustering-based strategies, or pattern-based approaches [Maletic and Marcus, 2000].

In most of the ensemble-based noise detection methods, majority or consensus filtering have been used. In majority filtering an instance is identified as being noisy if more than of 50% of classifiers in the ensemble misclassify it. Consensus filtering is a more restrictive approach, in which an instance has to be misclassified by all base learners to be marked as noise. In general, majority filtering outperforms consensus filtering. Consensus filtering is probably appropriate only for small ensembles, composed of a few predictors. In large ensembles the probability of identifying an instance as noise is typically small. Therefore, little or no cleansing occurs. The optimal noise detection strategy is problem dependent. Specifically, it depends on both the noise level and the type of base classifiers. In Chapter 4 of this thesis we will provide a detailed investigation of noise detection strategies that make use of ensembles.

As discussed earlier, an alternative strategy to deal with noise is by designing robust learning algorithms. Different strategies to improve the resilience to noise of these algorithms. The main idea is to determine how noisy instances affect the workings of a learning algorithm. Once the mechanisms by which noise hinders the performance of the algorithm one can try effect a redesign so as to avoid the detrimental effects of noise of the induced models. For instance, the size of the decision trees usually increases in the presence of noise [Quinlan, 1986]. Thus, pruning can be beneficial to improve the robustness of the decision tress [Breiman et al., 1984; Brodley and Friedl, 1999].

A further effective approach that can be beneficial in building a robust bootstrap ensemble is subsampling. Subsampling can be used as a regularization technique in ensemble methods. In chapter 3, we use this strategy in the bootstrapping ensemble methods, such as bagging, to improve its robustness.

2.5 Conclusions

In this chapter, we introduced the background topics on ensemble learning methods, with particular stress on classification techniques and learning in the presence of noise. Learning from a dataset consists of extracting patterns from the training data that can be applied to make predictions on the new examples. Specifically, for classification approaches, we aim to predict a class label from a set of features. Choosing a suitable learning algorithm in an application, is a challenging task, due to the different background and limitations of the learning algorithms and it depends upon many factors and in particular on data characteristics, such as the available amount of the data, data types, data quality, etc. Nevertheless, by using ensemble learning methods, one can mitigate the learning algorithms limitations by combining several predictions. The ensemble combination can consist of base learners generated using manipulations in the

same learning method or different learning methods, which refers to homogeneous and heterogeneous ensembles, respectively. In this section we described some of the most important ensemble methods and their characteristics. Classification margin is one of the well-studied topics in machine learning which has been discussed closely related to the generalization ability of the learning algorithm. In this chapter, we also have discussed the concept of the margin in classification and in particular in ensemble learning context. Additionally, in this chapter, a brief explanation on model selection via Structural Risk Minimization and the general capacity of the learning models has been explained.

Finally, this chapter terminates with introductory concepts on class label noise. Different types of class label noise based upon their statistical properties are described and two general strategies to deal with class label noise in classification tasks are described as preprocessing and robust classification strategies.

Chapter 3

Small margin ensembles can be robust to class-label noise

3.1 Abstract

Subsampling is used to generate bagging ensembles that are accurate and robust to class-label noise. The effect of using smaller bootstrap samples to train the base learners is to make the ensemble more diverse. As a result, the classification margins tend to decrease. In spite of having small margins, these ensembles can be robust to class-label noise. The validity of these observations is illustrated in a wide range of synthetic and real-world classification tasks. In the problems investigated, subsampling significantly outperforms standard bagging for different amounts of class-label noise. By contrast, the effectiveness of subsampling in random forest is problem dependent. In these types of ensembles the best overall accuracy is obtained when the random trees are built on bootstrap samples of the same size as the original training data. Nevertheless, subsampling becomes more effective as the amount of class-label noise increases.

3.2 Introduction

The success of large margin classifiers [Freund, 2009; Friedman et al., 2000; Mason et al., 1999b; Smola and Bartlett, 2000] has prompted many researchers to posit that large margins are a key feature in explaining the effectiveness of these methods. In the context of ensembles, the margin is defined as the weighted sum of votes for the correct class minus the weighted sum of votes for the most voted class other than the correct one. The effectiveness of boosting has been ascribed to the fact that it produces large

margins on the training data. The margins increase as the ensemble grows because of boosting's progressive focus on instances that are difficult to classify [Schapire et al., 1998a]. Nonetheless, several empirical studies put in doubt the general validity of this view [Breiman, 1998; Mease and Wyner, 2008]. Furthermore, efforts to directly optimize the margin (or the minimum margin) have met with mixed results [Rätsch, 2001; Rätsch and Warmuth, 2002]. In contrast to boosting, bagging [Breiman, 1996c], random forest [Breiman, 2001] and class-switching [Breiman, 2000; Martínez-Muñoz and Suárez, 2005] ensembles do not tend to increase the classification margins. In this paper we show that subsampling can be used to generate bagging ensembles that are robust to class-label noise in spite of having small margins. By contrast, the effectiveness of subsampling in random forest is strongly problem dependent. Nevertheless, for both types of ensembles, subsampling becomes more effective as the amount of class-label noise increases.

As discussed in [Frénay and Verleysen, 2014; Zhu and Wu, 2004a], class-label noise is generally more harmful for classification accuracy than noise in the feature values. Therefore, it is important to design classifiers that are robust to errors in the class labels of the training instances. The deterioration in performance caused by this type of noise is mainly due to an increase of the variance of the classifiers [Abellán and Masegosa, 2010; Melville et al., 2004; Pölitz and Schenkel, 2012]. Bagging is robust to class-label noise because it is a variance reduction technique. As a result of its adaptive nature, boosting reduces the classification bias as well as the variance [Bauer and Kohavi, 1999; Webb, 2000]. However, the excessive emphasis on incorrectly labeled examples makes standard boosting algorithms ill-suited for handling this type of noise. Nonetheless, it is possible to design robust versions of boosting to address this shortcoming [Freund, 2009; Rätsch, 2001].

A bagging ensemble is a collection of classifiers whose predictions are combined by majority voting. Each of the classifiers in the ensemble is built on a different bootstrap sample from the original training data. In standard bagging, bootstrap samples of the same size of the original training set are used to build the individual classifiers. However, this prescription need not be optimal. Several empirical studies have shown that the generalization capacity of bagging can significantly improve when smaller bootstrap samples are used [Hall and Samworth, 2005; Martínez-Muñoz and Suárez, 2010; Zaman and Hirose, 2009]. Subsampling generally makes bagging more robust to label noise [Sabzevari et al., 2014]. The key to this improvement is how smaller sampling ratios affect isolated instances. By an isolated instance we mean one that is located in a region where the majority of neighboring instances belong to a different class. Assume a sampling ratio such that the bootstrap samples used to build the individual classifiers contain less than 50% of the original training instances. This means that each instance is present in less than half of the ensemble classifiers. Therefore, the decision on the

label of a given instance is dominated by classifiers trained on bootstrap samples that do not contain that particular instance [Hall and Samworth, 2005; Martínez-Muñoz and Suárez, 2010]. If the instance in question is an isolated one, it is likely to receive the class label of its neighbors (i.e. the local majority class). If the noise is uniform, most of the incorrectly labeled instances are far from the classification boundaries. They can therefore be viewed as isolated instances. In such cases, using smaller sampling ratios reduces the influence of these isolated noisy instances. Consequently, the ensemble becomes more robust.

In summary, this article presents a comprehensive empirical assessment of the accuracy and robustness of bagging and random forest ensembles as a function of the bootstrap sampling ratio. This study extends our previous work [Sabzevari et al., 2014] including more datasets, algorithms and experiments. In addition, we illustrate how small margin ensembles can be resilient to class-label noise.

The article is organized as follows: Section 3.3 reviews previous work on label noise, focusing on classification ensembles. Section 3.4 is devoted to exploring the relation between margin and accuracy for different bootstrap sampling ratios and noise levels. In section 3.5 we present the results of an extensive empirical evaluation of the performance of bagging and random forest ensembles built using subsampling. The experiments are carried out in a wide range of classification tasks with different amounts of class-label noise. Finally, the conclusions of this investigation are summarized in section 3.6.

3.3 Related work

Poor data quality and contamination by noise are unavoidable in many real-world classification problems [Frénay and Verleysen, 2014; Zhu and Wu, 2004a]. This has a strong potential to mislead the learning algorithms used for automatic induction from these data. Two types of noise can be present in these problems: class-label noise and polluted feature values [Frénay and Verleysen, 2014; Zhu and Wu, 2004a]. Class-label noise is the consequence of incorrect manual labeling, missing information or failures in the data measuring process. Feature noise is often the result of a faulty data gathering process [Frénay and Verleysen, 2014; Zhu and Wu, 2004a]. Class-label noise typically has a more pronounced misleading effect than feature noise, except when most of the feature values are corrupted [Zhu and Wu, 2004a]. Frénay et al. [Frénay and Verleysen, 2014] identify three types of label noise, characterized by different statistical models: The Noisy Completely at Random Model (NCAR), in which the probability of a class-label error is independent of the values of the features, the actual class of the instance and the noise rate. To simulate this type of noise the class labels of randomly selected instances

are changed to a different class label, also at random. The second model is Noisy at Random (NAR). Labelling errors in this model are assumed to occur with a different probability for each class. NAR is useful to characterize tasks in which some classes are more susceptible to mislabeling than others. The third model is Noisy Not at Random (NNAR). In this case, the probability of an error depends on the actual class label and on the values of the features. This model should be used when some regions of the feature space, such as boundaries or sparse regions, are more prone to noise than others. Noise can be handled in a preprocessing step (data cleansing) or during the learning process, assuming that the algorithms used for induction from the contaminated data are robust [Frénay and Verleysen, 2014].

3.3.1 Data cleansing

To mitigate their harmful effects, noise and outliers can be eliminated in a preprocessing step, before the selected learning algorithm is applied. For instance, it is possible to use statistical models or clustering-based methods to detect outliers. Patterns and association rules can also be used in the cleansing process [Maletic and Marcus, 2000]. An example of a pattern-based data cleansing algorithm is described in [Segata et al., 2009, 2010]. In this method, local SVM's are used to identify and remove instances that are suspected to be noise. For each particular training instance, k-NN is applied to locate nearby instances. A SVM is then trained on these instances to find the optimal separating hyperplane in that neighborhood. If the label predicted by this locally trained SVM does not coincide with the actual label, the instance is identified as noisy and discarded. This cleansing method has been tested on real and artificial datasets, where it showed improvements over k-NN. In [Young et al., 2013], noisy instances are removed based on wrappers of different classification methods. In this study, the best results were obtained by removing or cleaning instances based on the prediction of a SVM built with the rest of the training data. Noisy instances are often included in the set of support vectors by a SVM classifier. Based on this observation, Fefilatyev et al. [Fefilatyev et al., 2012] propose to manually remove support vectors that are identified as noise by an expert. Then, a new SVM is built on the cleansed dataset. This process is iterated until no more support vectors are identified as noisy instances.

3.3.2 Robust learning algorithms

Another strategy to deal with noise is the design of robust learning algorithms. For instance, pruning is used in decision trees to reduce overfitting: The presence of noise tends to increase the size of the decision trees induced from the contaminated training

data. Pruning is thus an effective way to improve the robustness of decision trees [Breiman et al., 1984; Brodley and Friedl, 1999]. Another robustifying strategy is to explicitly incorporate in the learning algorithm the fact that the values of the features and the class labels can be polluted by noise. This strategy is adopted in the construction of Credal Decision Trees [Mantas and Abellán, 2014]. These types of trees are grown using the Imprecise Info-Gain Ratio (IIGR) as a splitting criterion. In this method the values of the features and class labels are approximated using probabilities and uncertainty measures.

It is also possible to adapt the algorithms used to build Support Vector Machines to improve their robustness to class-label noise. For instance, in [Stempfel and Ralaivola, 2009] the hinge loss is replaced by a related loss function that takes into account the amount of noise in the data. With this loss function the optimization problem becomes non-convex. Heuristic optimization methods are then used to search for the global minimum of this non-convex problem. Promising results were obtained by this robust SVM in problems with asymmetric class noise (NAR model). A drawback of this method is that it is necessary to estimate the amount of noise in the data. Another robust version of SVM, called P-SVM (Probabilistic SVM) is proposed in [Niaf et al., 2014] to classify magnetic resonance medical images. The P-SVM takes as inputs not only class labels but also class probability estimates. These probabilities are used to estimate the confidence on the labeling of each instance. The lower the confidence on the label, the lower the weight of that instance in the learning process. A practical limitation of this method is that one needs both qualitative (class labels) and quantitative (class posterior probabilities) information on the classes.

The problem of induction from noisy data has also been extensively addressed in the area of ensemble learning. In [Ali and Pazzani, 1994], Ali and Pazzani analyze the behavior of multiple classifier systems in the presence class-label noise. They observed that the improvements of the ensemble with respect to a single learner are generally smaller when the training data are contaminated with class-label noise. However, the reduction is not uniform and depends on the type of ensemble used.

Noise is not always harmful. In fact, noise injection is a powerful regularization mechanism that has the potential of improving the generalization capacity and robustness of prediction systems. In particular, randomization is used to build diverse ensembles that have good generalization capacity [Bauer and Kohavi, 1999; Breiman, 2000, 2001; Dietterich, 2000a; Frank and Pfahringer, 2006; Martínez-Muñoz and Suárez, 2005; Martínez-Muñoz et al., 2006, 2008; McDonald et al., 2003; Melville et al., 2004; Opitz and Maclin, 1999; Williams, 2011]. Furthermore, randomized ensembles, such as bagging and random forests, have been shown to be robust classifiers. By contrast, adaptive

ensembles, such as boosting, are very sensitive to class-label noise [Bauer and Kohavi, 1999; Dietterich, 2000a; McDonald et al., 2003; Melville et al., 2004; Opitz and Maclin, 1999]. The differences between these two types of ensembles can be explained by how errors are handled during the training phase: In bagging and random forest, the randomness injected during the construction of the ensemble is not correlated with the noise. For this reason, the influence of the different instances is equalized during training process [Grandvalet, 2004]. By contrast, boosting increases the weights of misclassified instances irrespective of whether they are correctly labeled or not. The emphasis on correctly labeled instances that are difficult to classify is beneficial, because it reduces the classification bias. However, the focus on outliers tends to mislead the learning process. The adaptivity that makes boosting such a powerful learner also renders it overly susceptible to noise.

There are many proposals to improve the robustness of boosting to class-label noise. In most of these variants the weight update rule is modified to reduce boosting's sensitivity to noise. A successful strategy is to use less aggressive weight updates. In standard boosting the weight updates are exponential. Using slower updating scheme moderates the emphasis on misclassified instances. This is generally advantageous because some of the misclassified instances could be outliers [Friedman et al., 2000]. In BrownBoost [Freund, 2001] misclassified instances with small negative margins are assigned higher weights, as in Adaboost. By contrast, instances whose margin is negative and above a specified threshold receive lower weights. The rationale behind this weight updating strategy is that instances in regions with a large class overlap tend to have low margins. By emphasizing these instances it is possible to model the classification boundary in more detail. Large negative margins correspond to isolated instances, which are far from the classification boundary. These instances are likely to be outliers and should therefore be discarded. In [McDonald et al., 2003], Brownboost is shown to be more robust than Adaboost in a limited experimental setting (5 datasets for 20% class-label noise). Another way of avoiding excessive emphasis on misclassified instances is to discard instances whose weight is above a threshold [Karmaker and Kwek, 2006]. The value of the threshold can be determined using a validation set. This algorithm is shown to be more robust than standard Adaboost in 8 datasets with low-medium class-label noise (up to 10%). None of these studies [Karmaker and Kwek, 2006; McDonald et al., 2003] compares the results of robust boosting ensembles with bagging. Finally, it is possible to combine bagging and boosting strategies to improve the accuracy and robustness of the resulting ensembles [Krieger et al., 2001; Webb, 2000]. However, as far as we are aware, the effectiveness of these hybrid ensembles have not been systematically evaluated in experiments with class-label noise.

In [Abellán and Masegosa, 2010] the authors propose to use credal decision trees to improve bagging's resilience to label noise. The results obtained with these types of ensembles in the low to medium noise regime (0%-10% class-label noise) are comparable to bagging of C4.5 trees. For higher noise levels (20%-30%) bagging of credal trees is more accurate than bagging of C4.5 trees.

Subsampling can also be used to design robust bootstrap ensembles. The individual classifiers of a bagging ensemble are built by applying the same base learning algorithm to different m-out-of-n bootstrap samples from the original training data. In standard bagging the number of instances in the bootstrap sample, m, is equal to the number of instances in the original training data, n (i.e. $m = n$). This choice of m need not be optimal. As an illustration, the performance of bagged nearest neighbors is comparable to the nearest neighbor algorithm itself [Breiman, 1996c]. However, if each bootstrap sample contains on average less than 50% distinct instances from the training set, the accuracy of bagged nearest neighbors can actually improve. In fact, if the sampling ratio tends to 0 as the training set size tends to ∞, the performance of bagged nearest neighbor tends to the Bayes (optimal) error [Hall and Samworth, 2005]. Another study [Zaman and Hirose, 2009] shows that subbagging with low sampling ratios generally improves the accuracy of bagging when stable classifiers are combined. The optimal subsampling ratio can be effectively determined using out-of-bag data [Martínez-Muñoz and Suárez, 2010]. Subsampling has also been shown to improve the robustness of bagging to class-label noise in some classification problems [Sabzevari et al., 2014]. In the current article, which is an extension of this work, we present the results of a comprehensive empirical study that provide further evidence of such improvement.

A comparison of the effectiveness of these different methods cannot be done on the basis of published results. For instance, the SVM's described [Niaf et al., 2014; Stempfel and Ralaivola, 2009] are tested in very specific cases: asymmetric noise [Stempfel and Ralaivola, 2009] or data in which class probabilities are available [Niaf et al., 2014]. An extensive empirical comparison of the different robust learning methods would be of great interest in the field. In terms of computational effort, ensembles of decision trees can be built faster than SVMs, at least in principle. Depending on the characteristics of the problem, the time complexity of SVM's is between quadratic and cubic in the number of training instances [Bottou and Lin, 2007]. Decision trees are faster to build: their time complexity is log-linear in the number of training instances and linear in the number of attributes [Witten et al., 2011]. The time needed to combine the individual decisions increases linearly with the number of base learners in the ensemble.

3.4 Subsampling in ensembles for noisy classification problems

In this section we explore how subsampling affects the classification margins in ensembles. The goal is to understand the relation between ensemble diversity, margins and robustness. We first present the results of a set of experiments that illustrate the effect of subsampling on the classification margin. Then we analyze how subsampling can act as a regularization mechanism that reduces the influence of mislabeled data.

3.4.1 Subsampling and margins

To understand how classification margins are affected by susbsampling we have carried out a series of experiments in the classification problems *Threenorm*, *Twonorm* and *Ringnorm* [Breiman, 1998]. These are synthetic datasets for which the optimum Bayes decisions are known. Bagging ensembles and random forests of 500 trees were trained using different bootstrap sampling ratios: 100%, which is the standard prescription, 20% and 5%. Ensembles trained on a noiseless set are used as a baseline. The bagging and random forest ensembles were built on the same training sets, which consist of 300 instances. The boosting ensembles were built on different sets of the same size. Additional ensembles were then built on copies of these sets contaminated with 20% label noise. The noise was simulated using the NCAR model. Bagging and random forest ensembles were tested using the out-of-bag error [Breiman, 1996b]. The out-of-bag data of a particular classifier consists of those instances which are not included in the bootstrap sample used to build that classifier. Since they are not used for training, they can be employed as independent test data. Thus, to compute the out-of-bag error, each instance in the training set is classified using only the votes of those predictors whose training sets do not include that particular instance. Besides providing a good estimate of the generalization capacity, the out-of-bag method allows us to analyze how the injected noise is handled by the ensemble: The same instances, including those whose class labels have been altered, are used both for training and for testing. To allow comparisons across ensembles, the performance of boosting was evaluated on the training data used to build the bagging and random forest ensembles.

Scatter plots of the posterior probability of class 2 versus the fraction of class 2 votes for the instances in the evaluation set are given in figures 3.1 and 3.2. The results displayed correspond to experiments with the different ensembles, sampling ratios and class-label noise levels. In bagging and random forest, the fraction of class 2 votes for a particular instance is estimated using the classifiers for which that instance was in the out-of-bag

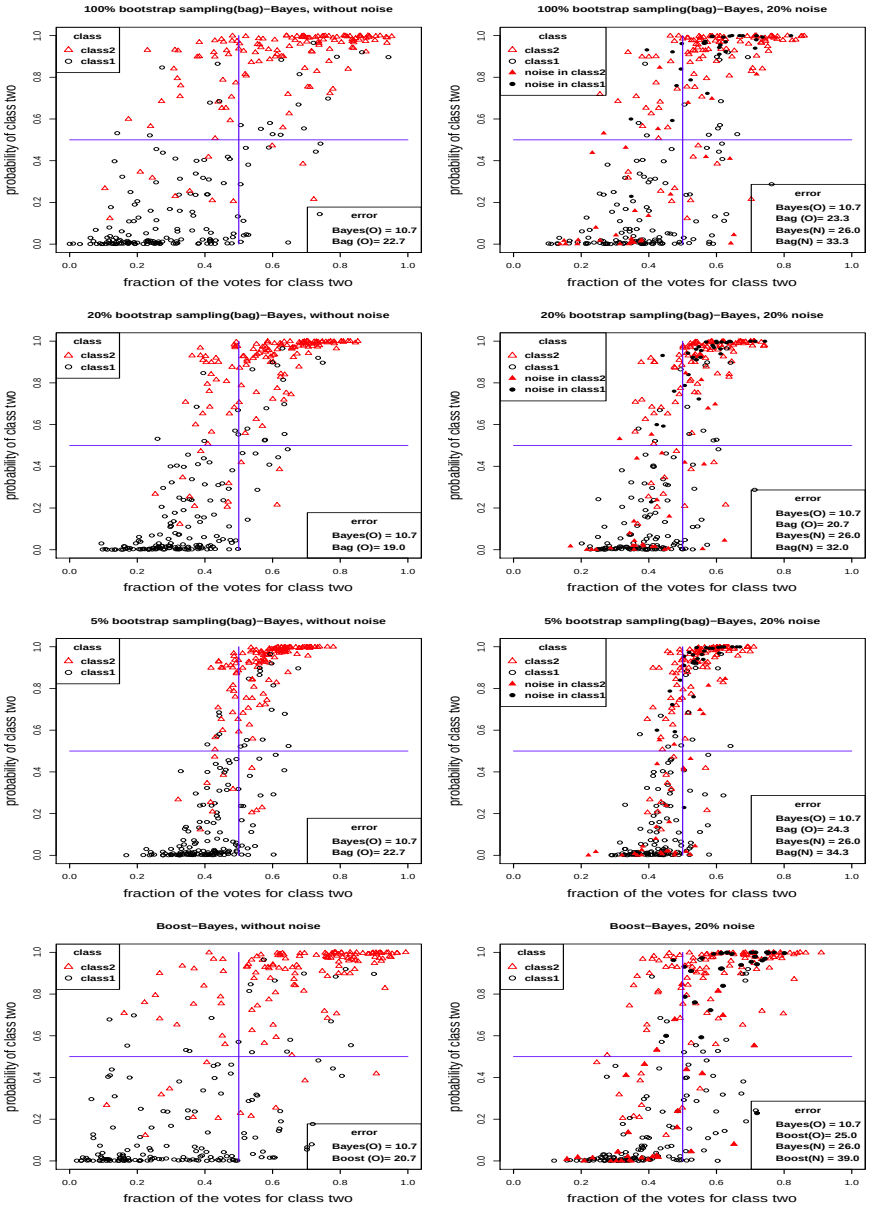


FIGURE 3.1: Scatter plots of the posterior probability of class 2 versus the fraction of ensemble class 2 votes for each instance in the evaluation set. Results are given for *Threenorm* without noise (left column) and with 20% noise (right column). The plots correspond to bagging ensembles with sampling ratios: 100% (first row), 20% (second row) and 5% (third row). The results for boosting are presented in the forth row.

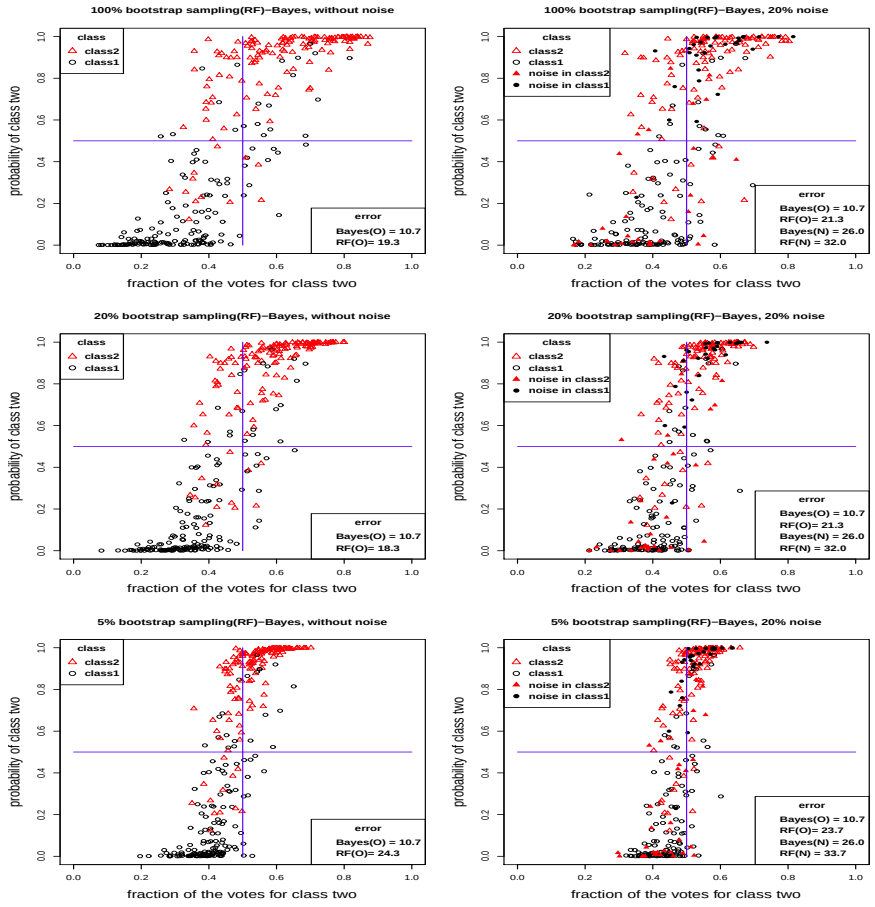


FIGURE 3.2: Scatter plots of the posterior probability of class 2 versus the fraction of ensemble class 2 votes for each instance in the evaluation set. Results are given for *Threenorm* without noise (left column) and with 20% noise (right column). The plots correspond to random forest ensembles with sampling ratios: 100% (first row), 20% (second row) and 5% (third row).

set (i.e. the set of instances not used to train that particular classifier). For boosting, all the classifiers in the ensemble were used. Figure 3.1 presents the scatter plots for an execution of *Threenorm*. Similar results are obtained in the other datasets. The plots included in this figure display (by rows) the results for standard bagging (100% sampling ratio), bagging using 20% sampling, 5% sampling and boosting. The results for a noiseless training set are presented in the first column. The results for a training set with 20% injected label noise are presented in the second column. Figure 3.2 shows the corresponding plots for random forest. In all plots the class 1 (class 2) instances are marked as empty circles (triangles). The instances whose class has been changed into class 1 (class 2) are marked as filled circles (triangles). The lines shown in the plots define the decision boundaries for the Bayes classifier (horizontal line) and the ensemble (vertical line). In addition, the errors for the ensembles and the Bayes classifier are displayed on the right bottom corner of the plots. For the problems with injected label

noise, error values considering noise (N) and without noise (O) are given. The Bayes classifier and the ensembles agree in the classification of instances located in the upper right and bottom left quadrants. The ensemble and the Bayes predictions are different for the remaining instances.

Several noteworthy features are revealed in these plots. In the noiseless problem (left column), the Bayes classifier assigns fairly high margins to most instances. The classification margins of bagging ensembles are lower than those of the Bayes classifier. Furthermore, they become smaller as the sampling ratio decreases. However, bagging ensembles with sampling ratios of 20% (second row) are more accurate than standard bagging, with 100% sampling (first row), in spite of the fact that the margins are smaller. The accuracy obtained with a sampling ratio of 5% is comparable to standard bagging. This is contrary to the view that accuracy should improve with increasing margin. A possible explanation of this behavior is that the different bootstrap samples have fewer common instances as the sampling ratio decreases. In consequence, the base classifiers become more diverse. This increased diversity initially leads to accuracy improvements. However, if the sampling ratio is reduced beyond a threshold, the individual classifiers become inaccurate. The error reduction that results from the aggregation of their decisions in the ensemble is not sufficient to compensate the lack of accuracy of the base learners. As a result, the fraction of instances with small and negative margins increases (see 5% sampling, third row, left plot).

A similar behavior is observed when label noise is present in the training set (right column): The classification margins are now smaller in all cases, relative to the noiseless situation. The test error (second row, right column) initially improves with decreasing sampling rates. However, if the sampling ratio is too low the performance of the ensemble eventually deteriorates. A similar behavior has been reported in class-switching ensembles [Martínez-Muñoz and Suárez, 2005].

The behavior for boosting (last row) is somewhat different. Because of its adaptive nature, boosting produces larger margins than bagging. While this is effective in the noiseless setting, it can be disruptive in noisy problems. In particular, when 20% class-label noise is injected boosting has the worst accuracy.

The results for random forest (shown in Figure 3.2) are qualitatively similar to those of bagging. However, the margins in random forest ensemble are typically smaller than in bagging or boosting. This is a consequence of the higher diversity provided by the random trees that make up the ensemble. From the experiments performed in this study the best overall results are achieved by random forests built with the standard 100% sampling ratio. The larger initial diversity of random forest implies that there is less room for improvement as the sampling ratio decreases. The variability introduced by

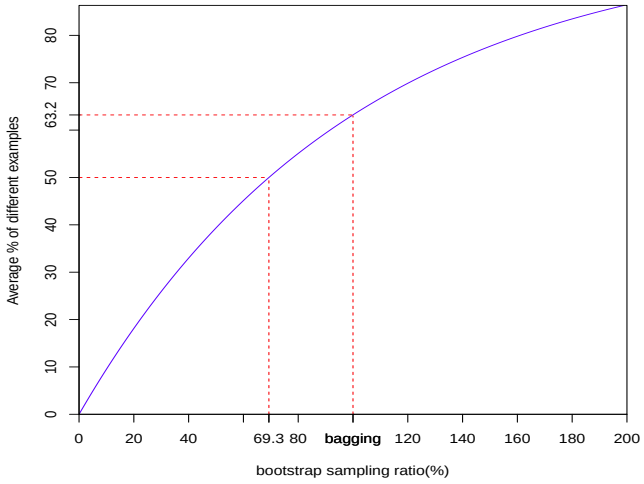


FIGURE 3.3: Average percentage of the unique training instances with respect to the size of the bootstrap sample

subsampling could in fact be detrimental to the accuracy of the ensemble. Therefore, subsampling is in general not as effective in random forest as it is in bagging. The validity of this qualitative analysis is confirmed by the empirical evidence presented in the section on experiments.

3.4.2 Subsampling as a regularization mechanism

Another way to understand how the sampling ratio can influence the performance of bagging ensembles is to consider the average number of distinct instances in each bootstrap sample. The dependence of this value with the sampling ratio is displayed in Figure 3.3. In standard bagging (100% sampling ratio) each bootstrap sample contains on average 63.2% different instances from the original training data [Breiman, 1996b]. The remaining 36.8% are repeated instances. As the sampling ratio becomes smaller, the number of distinct instances in each bootstrap sample decreases. Eventually, only one instance is sampled for a sampling ratio of $1/N$, where N is the size of the training set. The classifier built on such a sample would predict the class label of the single instance in the sample. Hence, the ensemble decision would be the majority class in the training data, irrespective of the values of the features. On the other extreme, bootstrap samples obtained with high sampling ratios contain most of the training instances. In such cases most base learners are very similar; the diversity arises only from having different repeated examples in the different bootstrap samples. Ensembles built using these extreme values of the sampling ratio will not in general have good generalization. The optimal performance is generally obtained at intermediate values of the sampling

ratio [Martínez-Muñoz and Suárez, 2010]. Furthermore, the optimal sampling ratio need not coincide with the standard prescription (100%).

An interesting regime corresponds to sampling ratios smaller than 69.3% (see Figure 3.3). For values below this threshold, fewer than 50% of the original training instances are included in each bootstrap sample. This means that each instance is present in less than half of the classifiers of the ensemble. In this regime, the class label given by the ensemble for each training instance is strongly influenced by the class label of nearby instances. In consequence, subsampling has the potential to increase the diversity of the classifiers in the ensemble. Higher diversity results in more variability in the votes and therefore in lower margins. We conjecture that using sampling ratios in this regime is an effective strategy to handle class-label noise in classification ensembles.

3.5 Experimental evaluation

In this section we presents the results of an empirical investigation of the performance of bootstrapping ensembles in the presence of label noise. The experiments are designed to assess how different sampling ratios affect the robustness of such ensembles. A total of 25 datasets from the UCI repository [Bache and Lichman, 2013] and other sources [Breiman, 1998] are used. They include synthetic data (*Ringnorm*, *Twonorm*, *Threenorm* and *Tic-tac-toe*) and classification problems from different application domains. The characteristics of the datasets are summarized in Table 3.1. They have been selected to cover a wide spectrum: there are problems with high and low numbers of attributes (e.g. *Sonar* and *Balance*, respectively), with small and large number of instances (e.g. *Magic04* and *Lung Cancer*, respectively), and with different numbers of classes.

The protocol used in the experiments is similar for all datasets. The only difference is in the generation of the training and test sets. For the synthetic datasets (*Threenorm*, *Ringnorm* and *Twonorm*) we generate a training set of 300 instances and a test set of 2000 instances. For the remaining datasets, 2/3 of the available data are used for training and 1/3 for testing. Stratified sampling is used to guarantee that the class distributions in the training and test sets are similar to the complete dataset. For each problem and realization of the training and test sets, the following steps are carried out:

1. Label noise is injected in the training set with different rates: 0% (no noise), 5%, 10% and 20%. In each case the class label of the randomly selected training instances is changed to a different class, also at random. This corresponds to the Noisy Completely At Random noise (NCAR) model [Frénay and Verleysen, 2014].

Dataset	**Instances**	**Test**	**Attrib.**	**Classes**
Australian	690	230	14	2
Balance	625	198	4	3
Breast W.	699	233	9	2
Diabetes	768	256	8	2
German	1000	333	20	2
Heart	270	92	13	2
Hepatitis	155	51	19	2
Horse-Colic	368	122	21	2
Ionosphere	351	117	34	2
Iris	150	50	4	3
Labor	57	38	16	2
Liver	345	115	6	2
Lung Cancer	32	10	56	3
Magic	19020	6340	11	2
New-thyroid	215	143	5	3
Ringnorm	300	2000	20	2
Segment	2310	1540	19	7
Sonar	208	699	60	2
Threenorm	300	2000	20	2
Tic-tac-toe	958	319	9	2
Twonorm	300	5000	20	2
Vehicle	846	564	18	4
Votes	435	145	16	2
Waveform	300	5000	21	3
Wine	178	59	13	3

TABLE 3.1: Characteristics of the classification problems and testing method

Uniform noise was used to avoid making specific assumptions about the structure of the noise.

2. For each contaminated training set, six bagging ensembles composed of 500 unpruned CART (Classification And Regression Tree) trees [Breiman et al., 1984] were built. The bootstrap sampling ratios used are: 10%, 20%, 40%, 60%, 80% and 100% (standard bagging). The CART trees were grown until pure class nodes were obtained. No pruning was applied to the fully grown decision trees. Random forest ensembles were built on the same training sets using the different sampling ratios. Random forest is a bagging ensemble composed of random trees. In random trees the splits at the inner nodes of the tree are selected from those that involve only a subset of randomly selected features. The size of these subsets was set to the square root of the number of features for each dataset [Bernard et al., 2009].

3. The generalization performance of all ensembles is gauged using the error on the test set. To obtain comparable results across all the ensembles considered no noise was injected in the test set.

TABLE 3.2: Relative error change for bagging and random forest for the different levels of noise and sampling ratios. The reference value corresponds to standard bagging in the noiseless case (marked in boldface as **1.00±0.00** in the table).

	Noise	10	20	40	60	80	100
Bag	0	1.38±1.43	1.06±0.41	0.98±0.16	0.96±0.08	0.98±0.08	**1.00±0.00**
	5	1.41±1.58	1.11±0.64	1.05±0.27	1.08±0.25	1.10±0.23	1.18±0.25
	10	1.45±1.70	1.19±0.92	1.13±0.43	1.18±0.44	1.28±0.47	1.38±0.57
	20	1.55±1.98	1.42±1.51	1.44±1.16	1.60±1.10	1.72±1.13	1.83±1.19
RF	0	1.77±2.85	1.49±2.24	1.21±1.44	1.06±0.91	0.97±0.58	0.94±0.42
	5	1.75±2.62	1.48±2.08	1.23±1.31	1.13±0.91	1.05±0.68	1.01±0.55
	10	1.77±2.56	1.48±1.98	1.27±1.36	1.20±1.03	1.15±0.82	1.16±0.76
	20	1.81±2.43	1.62±2.03	1.51±1.55	1.48±1.36	1.51±1.31	1.53±1.26

TABLE 3.3: Relative error change averaged over all datasets for bagging and random forest for the different levels of noise. The reference values are the test errors bagging and random forest noiseless case (marked in boldface in the first and fifth rows of the table).

	Noise	10	20	40	60	80	100
Bag	0	**1.00±0.00**	**1.00±0.00**	**1.00±0.00**	**1.00±0.00**	**1.00±0.00**	**1.00±0.00**
	5	1.01±0.09	1.03±0.14	1.06±0.14	1.12±0.19	1.12±0.20	1.18±0.25
	10	1.03±0.11	1.07±0.22	1.13±0.30	1.22±0.38	1.30±0.45	1.38±0.57
	20	1.08±0.13	1.23±0.44	1.43±0.95	1.66±1.02	1.75±1.12	1.83±1.19
RF	0	**1.00±0.00**	**1.00±0.00**	**1.00±0.00**	**1.00±0.00**	**1.00±0.00**	**1.00±0.00**
	5	1.05±0.18	1.02±0.08	1.05±0.08	1.09±0.16	1.08±0.11	1.06±0.11
	10	1.09±0.30	1.06±0.18	1.09±0.16	1.14±0.15	1.18±0.18	1.21±0.26
	20	1.18±0.51	1.22±0.39	1.35±0.37	1.44±0.39	1.55±0.49	1.59±0.53

The test errors reported in the tables are averages over the 100 realizations of the training and test sets.

3.5.1 Results

To give an overall view of the results, we have computed the averages of the test error changes in the 25 problems investigated, for each noise level, sampling strategy and ensemble method (bagging and random forest). The results are presented in Table 3.2 as the relative error change, using standard bagging in the noiseless setting as the reference value. This reference value is marked in boldface in the Table. Values below 1 indicate that, on average, the corresponding method outperforms standard bagging in the noiseless setting. Values above 1 signal a higher average test error.

In addition, the average error changes with respect to the noiseless setting for each ensemble type are shown in Table 3.3. The reference values are highlighted in boldface. These results serve to analyze how the accuracy of ensembles built with the different sampling ratios is affected by class-label noise. The average test error changes for the individual datasets are presented in the appendix: Tables 3.6, 3.7 and 3.8 for bagging and Tables 3.9, 3.10 and 3.11 for random forest ensembles.

An analysis of the results presented in Table 3.3 reveals that the loss of accuracy with respect to the noiseless setting is very different for different sampling ratios. For standard bagging with 20% noise injected, the average error increase with respect to the noiseless case is 83%. This large increase should be expected, given the high level of noise injected. By contrast, if a 10% sampling ratio is used, the average error increase is only 1.0%, 3.0% and 8.0% for the 5%, 10% and 20% label noise rates, respectively. An interesting observation is that these error increments are significantly lower than the corresponding levels of the noise that has been injected. Using lower sampling ratios in bagging tends to increase the variability of the base classifiers. This larger ensemble diversity generally translates into more robust classification. The remarkable robustness to class-label noise of these ensembles is illustrated in greater detail by the results presented in Tables 3.6, 3.7 and 3.8 in the appendix. In some cases, there is even an improvement in the classification accuracy when noise is injected. For instance,the best overall accuracy of bagging in *Breast* with 20% noise is achieved using a 10% sampling ratio: The test error goes from 4.1% when no noise is injected to 3.5% when the training data has 20% noise. By contrast, when standard bagging is used, the test error increases almost 5 percentage points (from 4.3% with no noise to 9.2% with 20% noise).

For random forest ensembles, a similar, albeit less marked effect, is observed in Table 3.3: The deterioration with the level of noise injected is more pronounced for larger sampling ratios (18% increment with a 10% sampling ratio and 59% with a 100% sampling ratio). However, the baseline accuracy of random forest ensembles at low sampling ratios is rather poor: In the noiseless setting, the average error rate of random forest with a 10% sampling ratio is 77% larger than standard bagging (see Table 3.2). One of the reasons why subsampling is not as effective is that random forests are typically more diverse than bagging ensembles. This diversity makes standard random forest more robust to noise (see rightmost column of Table 3.2). Using lower sampling ratios is not as effective in increasing the diversity of the random trees. Therefore, subsampling does not lead to systematic accuracy improvements in random forest ensembles.

Finally, from the analysis of the results displayed in Table 3.2 one concludes that the best overall performance in the noiseless setting is achieved using standard random forests (0.94). The difference with standard bagging is 6 percentage points on average. However, the difference between standard random forest and bagging using 60% sampling ratio is only of two percentage points (values 0.96 and 0.94 in Table 3.2). As the noise level increases the best overall accuracy corresponds to bagging using 20-40% sampling ratios (1.42 and 1.44 in the Table 3.2 for a 20% noise rate).

3.5.2 Accuracy as a function of ensemble size

The error curves displayed in Figures 3.4 and 3.5 trace the dependence of the average test error of bagging on the number of classifiers in the ensemble. The classification problems used to illustrate this dependence are *Australian* (Figure 3.4) and *Threenorm* (Figure 3.5). The curves displayed correspond to different sampling ratios and noise levels: noiseless setting (top left plot), 5% (top right plot), 10% (bottom left plot) and 20% (bottom right plot) noise rates. The qualitative features of these error curves are similar in all the classification problems investigated.

When no noise is injected, the error curves for *Australian* converge to their asymptotic (infinite ensemble) limit after approximately 50 trees. As more noise is injected larger sizes are required for convergence. In this dataset the qualitative behavior of the error as a function of ensemble size is similar for the different sampling ratios. By contrast, in *Threenorm* (Figure 3.5), the convergence of the ensemble error curves is slower for smaller sampling ratios.

3.5.3 Statistical significance of the results

A record of the statistically significant differences in accuracy with respect to the standard ensembles in the 25 classification problems investigated is given in Tables 3.4 and 3.5 for bagging and random forest, respectively. In each cell of these tables the number of times a given method wins, draws or looses against standard bagging (Table 3.4) or standard random forest (Table 3.5) is displayed. Paired t-tests with $\alpha = 0.05$ are used to determine the significant wins and losses. A draw is recorded if the differences between the test errors are not statistically significant.

Noise (%)	10%	20%	40%	60%	80%
0	9/5/11	15/3/7	13/8/4	13/12/0	1/23/1
5	11/6/8	17/2/6	16/6/3	14/11/0	7/18/0
10	15/5/5	19/2/4	17/7/1	13/12/0	7/18/0
20	20/2/3	21/1/3	18/6/1	14/11/0	9/16/0

TABLE 3.4: Records for statistically significant wins/draws/losses for bagging with subsampling for different sampling ratios with respect to standard bagging (100 % sampling ratio).

From the results presented in these tables one concludes that subsampling is more effective at higher levels of label noise. For instance, from Table 3.4, bagging using a 10% sampling ratio and 0% noise significantly outperforms standard bagging in 9 datasets and obtains lower accuracy in 11 datasets. When the noise rate is increased to 20%, the situation reverses: there are 20 wins and only 3 significant losses.

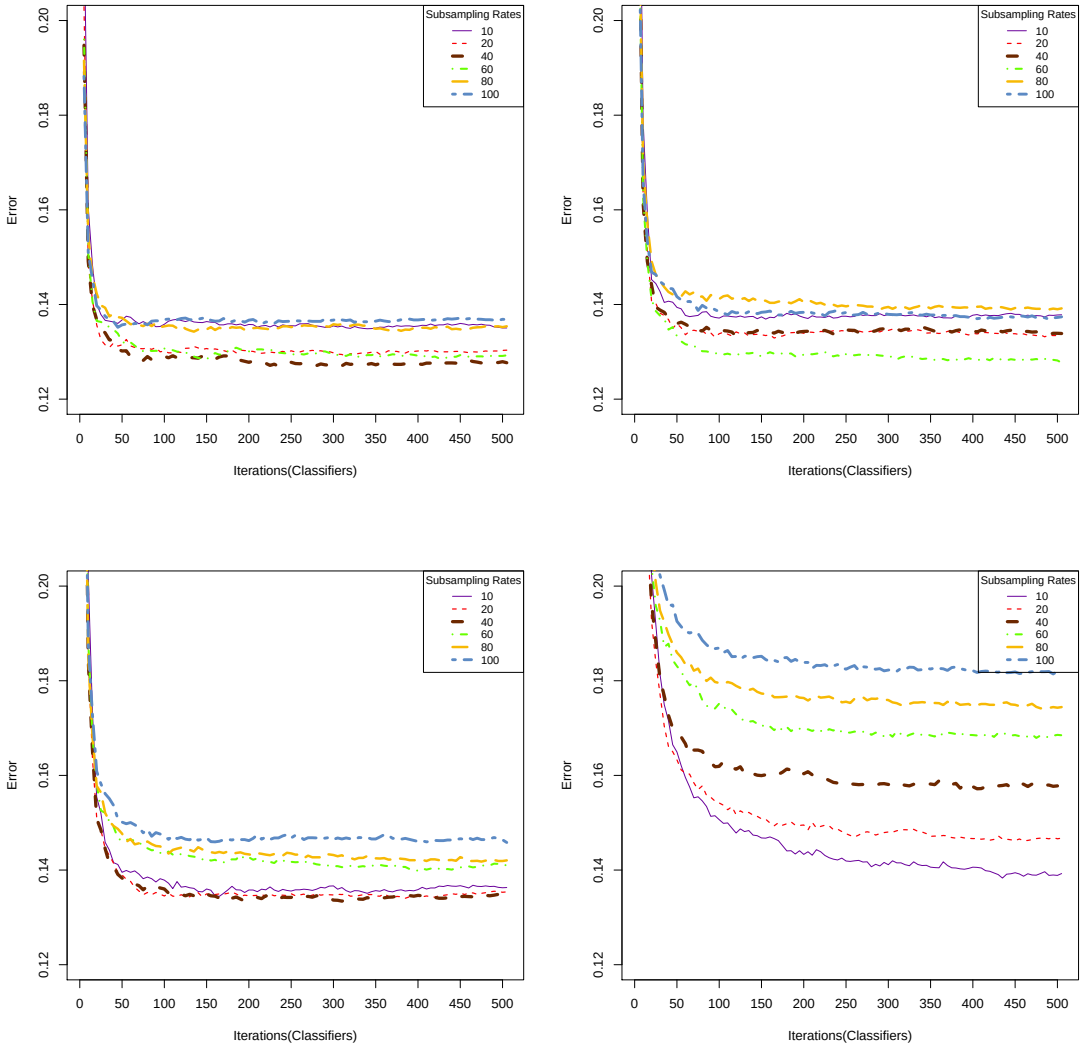


FIGURE 3.4: Average test error of bagging in the *Australian* dataset: Noiseless setting (top left); 5% (top tight), 10% (bottom left) and 20% (bottom right) noise rates. The different curves in each plot correspond to different sampling ratios.

An analysis of the results for random forest in Table 3.5 leads to similar conclusions. Subsampling becomes more effective also at lower sampling ratios. The effect, however, is less salient than in bagging. In the noiseless case random forest using 20% bootstrap sampling outperforms the standard version in only one dataset and losses in 21 datasets. When the noise rate is increased to 20% the number of wins increases to 11 and the number of losses decreases to 9. Random forests built using the standard prescription (100% sampling ratio) have the best overall performance in the problems investigated for all noise levels. However, as the amount of class-label noise increases, subsampling becomes more effective and is actually advantageous in some problems.

Finally, the method proposed by Demšar in [Demsar, 2006] is used to compare the

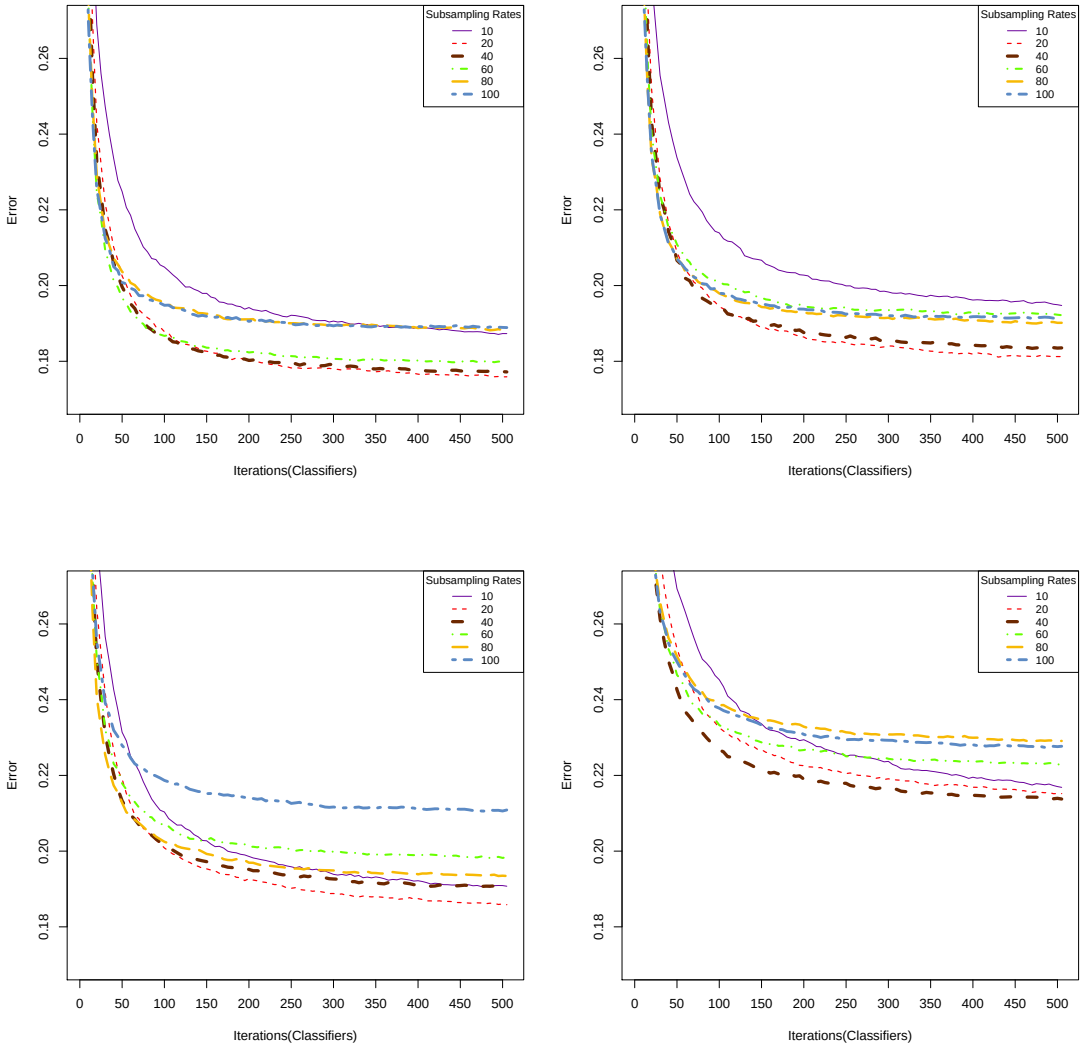


FIGURE 3.5: Average test error of bagging in the *Threenorm* dataset: Noiseless setting (top left); 5% (top tight), 10% (bottom left) and 20% (bottom right) noise rates. The different curves in each plot correspond to different sampling ratios.

performance of the ensembles across the different datasets. The comparison is made in terms of the average rank of each classifier in the problems considered. For a given dataset, the rank of the different ensembles is computed on the basis of the average test errors in the different realizations of the training and test sets. Figure 3.6 present the results of these tests for different noise levels and sampling ratios. A Nemenyi test with p-value<0.05 is used to determine the statistical significance of the differences between average ranks. The critical distance above which these differences are considered significant is shown for reference (CD $= 1.5$ for 6 methods, 25 dataset and p-value<0.05). In the diagrams, if two methods are connected with a horizontal solid line, the difference between their average ranks is not statistically significant.

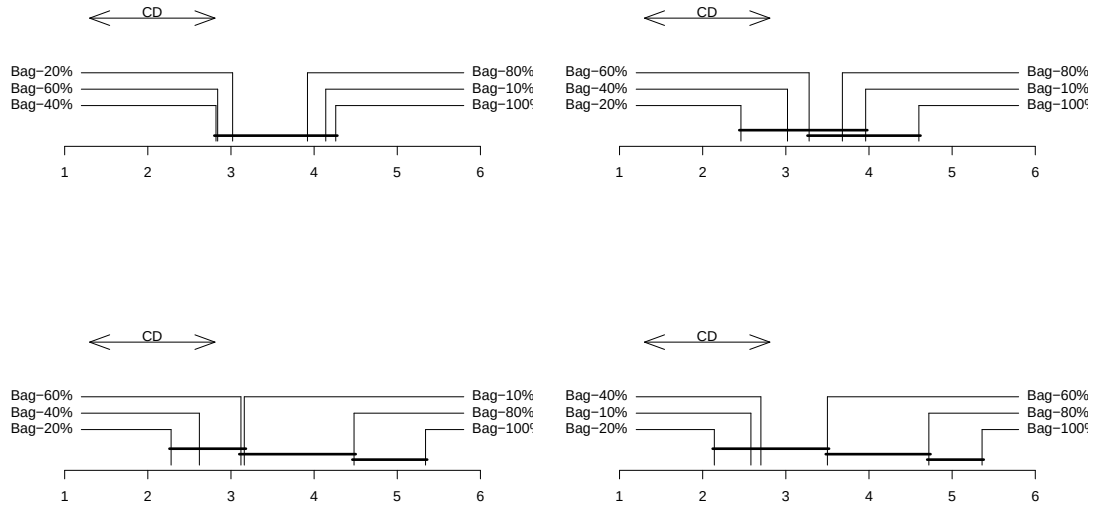


FIGURE 3.6: Comparison of bagging with different sampling ratios using the Nemenyi test, for datasets without noise (top left) and with 5% (top right), 10% (bottom left) and 20% (bottom right) noise rates. Horizontal lines connect sampling ratios whose average ranks are not significantly different (p-value < 0.05).

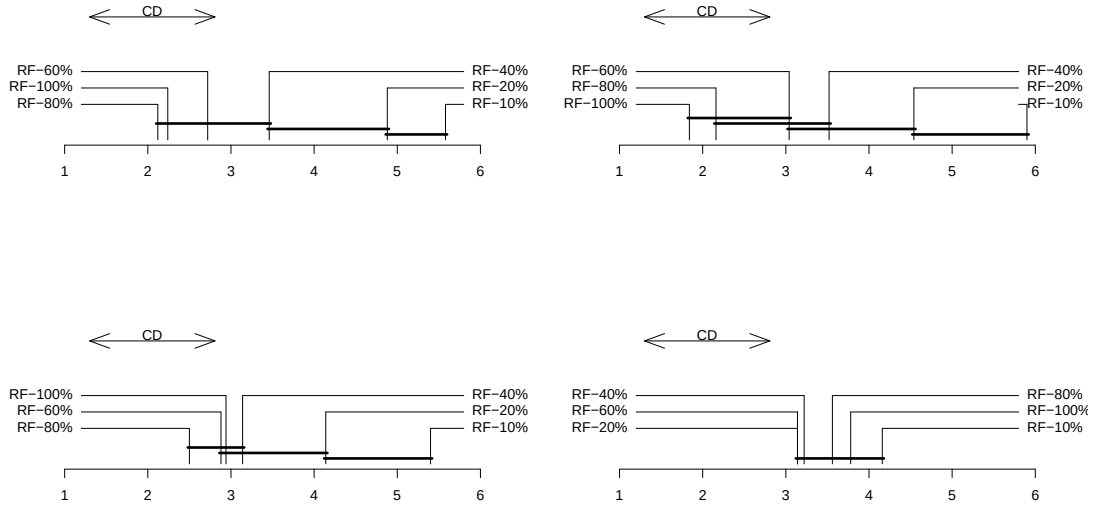


FIGURE 3.7: Comparison of random forest with different sampling ratios using the Nemenyi test, for datasets without noise (top left) and with 5% (top right), 10% (bottom left) and 20% (bottom right) noise rates. Horizontal lines connect sampling ratios whose average ranks are not significantly different (p-value < 0.05).

Noise (%)	10%	20%	40%	60%	80%
0	1/1/23	1/3/21	2/14/9	5/12/8	3/19/3
5	0/5/20	1/6/18	2/14/9	1/19/5	1/22/2
10	3/4/18	3/9/13	3/16/6	4/19/2	3/21/1
20	8/6/11	11/5/9	9/10/6	6/16/3	3/19/3

TABLE 3.5: Records for statistically significant wins/draws/losses for random forest with subsampling for different sampling ratios with respect to standard random forest (100 % sampling ratio).

Figure 3.6 displays the results of the Demšar test for bagging ensembles in the noiseless setting (top left), and with 5% (top right), 10% (bottom left) and 20% (bottom right) noise rates. In all cases, standard bagging with 100% sampling ratio has the lowest average rank. When no noise is injected the highest accuracy corresponds to bagging with 20%, 40% and 60% sampling ratios. However, the differences with other sampling ratios are not statistically significant. The improvements over standard bagging for 20% and 40% sampling ratios are statistically significant in the problems with noise rates 5%, 10% and 20%. For the 20% noise rate, bagging ensembles that use 10%, 20%, 40% and 60% sampling ratios are significantly better than standard bagging (100% sampling ratio).

The results of the Demšar test for random forest are displayed in Figure 3.7. Standard random forest (i.e. with 100% sampling ratio) is the best method for the noiseless setting (top left plot) and for 5% noise rate (top right plot). However, the differences with ensembles built with 80%, 60% and 40% sampling ratios are not statistically significant. For these noise rates standard random forest significantly outperform ensembles built using 20% and 10% sampling ratios. When higher noise levels are injected (10%), the best performance corresponds to random forest with 80% sampling ratio. The improvements over ensembles built with 10% and 20% sampling ratios are statistically significant. For the highest noise level (20%) the method with the highest average rank is random forest with a 20% sampling ratio. However, in this case, none of the differences between the average ranks of the different ensembles are statistically significant.

3.6 Conclusion

In this article we have analyzed the resilience to class-label noise of bootstrap aggregation ensembles as a function of the size of the bootstrap samples used to train the individual predictors. The results of an extensive empirical evaluation show that bagging composed of unpruned decision trees trained on bootstrap samples whose size is between 10% and 40% of the size of the original training set are more resilient to label noise than standard

bagging (i.e. using a 100 % sampling ratio). For random forests subsampling is effective only in noisy domains ($\approx 20\%$ noise in the class labels) and in specific classification tasks. In most problems, for low noise levels the best results are obtained using high sampling ratios. In fact, using the standard sampling ratio to build random forests is a reasonable choice with a good overall performance in the problems investigated, especially in the absence of class-label noise. However, in noisy tasks, it is worth to explore the possibility of subsampling, because the optimal size of the bootstrap samples is problem dependent.

Experiments in synthetic data have been used to illustrate that the classification margins become smaller as the sampling ratio decreases. This effect occurs both in the noiseless setting and when class-label noise is injected. They provide empirical evidence that using smaller bootstrap samples to build the individual ensemble classifiers can lead to improvements in accuracy, especially in noisy domains. However, if the sampling ratio decreases beyond a threshold the accuracy of the ensemble abruptly drops. This abrupt deterioration of performance occurs at higher sampling rates in random forests than in bagging. The reason is that the margins are typically larger in bagging than in random forests. Since lower sampling ratios tend to reduce the margin, the potential improvements of subsampling for random forest are realized only in problems with high levels of class-label noise.

Appendix

Tables 3.6, 3.7 and 3.8 display the average test error (with the standard deviation after the $\pm$ sign) of bagging for the different sampling ratios and noise rates considered. The results are presented in three separate tables for the sake of clarity. In each row the lowest error is highlighted with an asterisk (*). For each noise level and dataset (i.e. for each row), values that are significantly better than standard bagging (column 100%) are highlighted in boldface. Results that are significantly worse than standard bagging are underlined. The statistical significance of these differences is determined using paired t-tests at a significance level $\alpha = 0.05$. The corresponding results for random forest ensembles are presented in Tables 3.9, 3.10 and 3.11.

TABLE 3.6: Bagging average test error I

Dataset	Noise (in %)	Bootstrap sampling ratio 10%	20%	40%	60%	80%	100%
Australian	0	13.5±1.9	**13.0±1.9**	**12.8±1.6***	**12.9±2.0**	13.5±1.9	13.7±2.2
	5	13.8±1.9	13.3±1.9	13.4±2.0	**12.8±2.0***	13.9±2.0	13.7±2.0
	10	**13.6±1.9**	**13.5±2.2***	**13.5±1.9***	14.1±1.7	14.2±2.0	14.6±2.2
	20	**13.9±2.1***	**14.7±1.8**	**15.8±2.3**	**16.8±2.6**	**17.4±2.5**	18.2±2.6
Balance	0	**10.2±0.9***	**11.4±1.7**	**13.8±1.4**	**16.0±1.9**	17.4±1.8	18.3±2.1
	5	**11.0±1.1***	**11.9±1.3**	**14.7±1.2**	**17.1±1.6**	18.3±1.4	19.8±2.0
	10	**11.2± 1.2***	**12.9±1.5**	**16.3±1.6**	**17.9±1.4**	19.2±1.9	20.3±2.1
	20	**12.8±1.1***	**14.9 ±1.6**	**18.5±1.8**	**20.9 ±1.8**	23.6±3.0	25.1±3.2
Breast W.	0	4.1±1.3	**3.7±1.1***	**3.8±1.0**	**3.9±1.1**	4.1±1.0	4.3±1.1
	5	**3.7±1.0**	**3.5±1.0***	**3.7±1.0**	**4.2±1.3**	4.7±1.2	5.0±1.6
	10	**3.5±1.2***	**3.6±1.0**	**3.7±1.2**	**4.4±1.2**	**5.3±1.5**	6.1±1.7
	20	**3.5±1.0***	**4.2±1.3**	**5.1±1.4**	**6.4±1.7**	**8.0±2.1**	9.2±2.2
Diabetes	0	**23.7±2.2***	**23.8±2.1**	24.1±2.6	**23.8±1.9**	24.6±2.3	24.4±2.3
	5	**23.7±2.4***	**24.2±1.9**	**24.2±2.3**	**24.2±2.1**	24.8±2.2	25.2±2.3
	10	**23.4±2.2***	**24.2±2.4**	**24.6±2.1**	**25.1±2.5**	25.8±2.0	25.9±2.3
	20	**24.5±2.3***	**24.9±2.3**	**26.8±2.4**	**27.1±2.9**	**27.7±3.0**	28.5±2.5
German	0	25.0±1.8	24.2±1.6	23.9±2.0*	23.9±1.8*	24.0±1.8	24.3±1.8
	5	25.4±1.6	**24.1±1.8***	**24.3±1.9**	**24.2±1.7**	**24.5±1.9**	25.1±1.8
	10	25.5±1.6	**24.6±1.8***	**24.6±1.8***	25.4±2.1	25.8±2.0	26.0±2.3
	20	**26.5±1.8**	**25.8±1.9***	**26.4±2.1**	**27.6±2.5**	28.0±2.5	28.5±2.1
Heart	0	**17.0±3.5***	**17.1±3.7**	**18.7±4.4**	19.0±3.7	19.3±3.8	19.9±3.4
	5	**17.4±4.0***	**18.6±4.2**	**18.8±4.0**	**19.3±4.3**	**20.4±3.7**	21.8±4.6
	10	**18.1±3.9***	**19.1±4.6**	**19.5±3.7**	21.5±4.0	**21.2±4.4**	22.3±4.1
	20	**20.3±3.8***	**22.1±4.7**	**22.4±4.7**	**24.3±4.4**	**24.3±5.1**	25.9±4.3
Hepatitis	0	**21.2±0.4**	**19.7±2.0***	**19.8±1.9**	**20.7±2.7**	21.5±4.0	22.2±3.3
	5	**20.1±2.5**	**19.8±2.0***	**20.8±2.6**	**21.3±2.9**	**22.2±3.6**	23.3±4.1
	10	**20.0±2.6**	**19.8±1.8***	**21.1±3.1**	**22.4±3.7**	**23.5±4.4**	25.0±4.8
	20	**20.2±3.6***	**20.2±2.6***	**24.9±4.2**	**25.8±4.4**	**27.9±5.6**	31.4±5.2
Horse-Colic	0	25.2±2.1	19.9±0.9	**16.1±0.4***	**16.1±0.5***	17.2±0.7	16.4±0.9
	5	25.8±2.4	21.8±2.2	17.8±2.1	17.1±2.1	17.1±1.8	17.0±2.5*
	10	26.4±2.5	23.3±2.9	19.4±2.8	18.5±2.9*	18.5±3.2*	18.6±2.8
	20	27.5±3.8	25.8±3.9	22.4±3.5	22.4±3.8	21.3±3.6*	21.9±3.8
Ionosphere	0	9.6±2.8	**6.8±1.9***	7.5±2.2	**7.2±2.1**	7.7±2.5	8.0±2.4
	5	9.1±2.5	**7.2±2.3***	**7.6±2.3**	8.6±2.9	8.5±2.6	8.4±2.5
	10	9.6±2.2	**7.4±2.3***	**7.9±2.6**	**8.1±2.7**	9.1±2.7	9.6±2.8
	20	**10.3±3.1**	**9.8±3.2***	**10.1±3.0**	**11.2±3.5**	12.7±3.4	13.0±3.7
Iris	0	12.3±4.6	**4.5±2.6***	5.2±2.7	5.3±2.4	5.2±2.8	5.3±2.4
	5	8.6±6.0	**5.1±3.2***	**5.3±2.8**	**5.3±2.5**	7.4±3.3	7.9±3.7
	10	**4.6±6.8***	**4.7±3.4**	**5.3±2.6**	**6.4±3.4**	**8.9±4.0**	10.6±5.0
	20	**5.0±3.1***	**6.1±3.5**	**7.0±4.6**	**10.8±5.1**	**13.4±5.4**	16.0±6.2

TABLE 3.7: Bagging average test error II

Dataset	Noise (in %)	Bootstrap sampling ratio 10%	20%	40%	60%	80%	100%
Labor	0	16.2±8.8	14.7±8.5	13.3±9.8	11.8±7.6*	13.6±5.1	12.0±6.4
	5	16.0±10.2	13.3±8.8	14.2±8.8	12.4±8.5	11.8±7.0	10.4±5.6
*	10	14.2±7.2	11.8±6.3*	17.6±14.6	15.8±6.9	17.8±10.3	16.0±9.4
	20	18.9±8.3	17.3±9.0*	17.8±8.3	18.4±9.6	22.9±10.4	20.0±10.0
Liver	0	**28.6±4.0**	**27.4±3.4***	**27.5±3.7**	**27.8±3.4**	**28.7±3.6**	29.9±3.6
	5	**29.0±3.5**	**28.5±3.8***	**28.5±4.3***	**29.5±4.3**	30.1±3.9	30.7±3.9
	10	**29.9±4.0**	**29.1±3.4**	**29.0±4.0***	30.4±3.8	31.0±3.3	31.4±3.7
	20	**32.4±4.3**	**31.9±4.5***	**32.3±4.4**	**32.9±4.3**	34.3±4.1	34.8±4.3
Lung Cancer	0	**42.0±8.9***	53.5±10.2	**42.0±11.2***	45.5±11.1	45.5±11.8	45.0±12.9
	5	**44.0±8.5***	53.0±10.8	49.0±11.5	47.5±12.2	46.5±11.5	49.5±12.7
	10	**42.0±9.1***	50.5±10.4	47.0±11.7	45.5±12.0	49.0±11.9	49.0±11.8
	20	**49.5±9.0**	**53.5±11.1**	**48.0±12.0**	43.5±12.8*	55.0±12.2	54.0±13.7
Magic	0	13.0±0.4	12.5±0.4	12.3±0.3	12.3±0.4	12.2±0.4*	12.2±0.4*
	5	13.1±0.4	12.7±0.4	12.5±0.4	**12.3±0.3***	12.4±0.4	12.5±0.3
	10	13.0±0.4	12.8±0.3	**12.7±0.4***	**12.7±0.4***	12.9±0.3	12.9±0.4
	20	**13.4±0.4**	**13.2±0.4***	**13.3±0.4**	**13.6±0.4**	**13.8±0.4**	14.2±0.4
new-thyroid	0	5.4±3.0	6.4±2.9	6.9±3.2	5.2±3.2*	5.6±3.1	5.7±2.7
	5	6.5±2.5	**4.5±1.8***	**5.3±2.7**	6.7±3.0	**5.0±2.0**	8.3±2.8
	10	6.3±3.8	**5.4±2.8**	5.2±2.5	**5.0±2.3***	6.7±3.5	7.9±3.6
	20	**5.2±3.1**	**5.1±2.7***	**5.8±2.5**	10.1±4.4	11.0±3.6	10.6±5.4
Ringnorm	0	12.1±1.1	**8.1±1.1**	**7.6±1.3***	**8.2±1.8**	8.6±1.7	8.8±1.9
	5	11.4±1.7	**7.9±1.3**	**7.4±1.3***	**8.0±1.7**	**8.4±1.6**	9.1±1.8
	10	11.3±1.9	**7.8±1.5**	**7.5±1.5***	**8.4±1.6**	**8.7±1.6**	9.5±1.9
	20	11.5±2.1	**8.6±1.5***	**9.1±1.9**	**9.7±1.9**	**10.1±1.7**	11.2±1.9
Segment	0	3.4±1.4	3.0±1.2	2.6± 1.7	2.3± 1.5	2.2± 1.0	2.1± 0.9 *
	5	**3.2±1.5**	**3.1±1.3***	**3.4±1.9**	3.8±1.9	3.6±0.7	3.8±1.1
	10	**3.2±1.2**	**3.1±1.3***	**4.2±1.1**	4.6±1.7	5.2±1.2	6.6±1.3
	20	**3.5±2.1**	**3.2±1.5***	**4.0±1.6**	5.7±1.5	7.2±1.4	7.4±1.5
Sonar	0	22.5±4.4	23.6±4.3	23.0±4.6	21.5±4.9 *	22.0±4.7	21.0±4.9
	5	24.7±4.2	24.0±5.3	23.2±4.2	21.3±4.5*	22.4±4.6	22.7±5.3
	10	24.8±4.6	22.7±5.1	23.9±5.2	21.7±4.8*	24.1±5.4	21.8±5.0
	20	**25.6±5.1**	**25.7±5.5**	26.8±5.8	25.2±5.4*	26.3±5.9	26.2±6.0
Threenorm	0	18.7±1.2	**17.6±1.3***	**17.7±1.4**	**18.0±1.7**	18.8±1.6	18.9±1.8
	5	19.5±1.3	**18.1±1.6***	**18.4±1.4**	19.2±1.8	19.0±1.6	19.1±1.5
	10	**19.1±1.5**	**18.6±1.3***	**19.1±1.5**	**19.8±1.5**	**19.3±1.8**	21.1±1.7
	20	**21.7±1.9**	**21.5±1.9**	**21.4±1.8***	22.3±1.9	22.9±1.9	22.8±2.0
Tic-tac-toe	0	15.4±2.0	5.1±2.0	2.2±0.9	2.0±0.9	1.9±0.8*	1.9±0.7 *
	5	16.9±2.5	7.6±2.3	3.5±1.3	**3.1±1.2***	3.3±1.2	3.6±1.4
	10	18.0±2.3	10.4±2.1	5.4±1.7	**5.1±1.6***	5.4±1.6	5.6±1.6
	20	20.8±2.6	16.3±2.7	13.0±2.4	12.2±2.1*	12.2±2.1*	12.7±2.7

TABLE 3.8: Bagging average test error III

Dataset	Noise (in %)	Bootstrap sampling ratio 10%	20%	40%	60%	80%	100%
Twonorm	0	**4.9±1.1**	**4.6±0.8***	**5.1±1.0**	**5.2±0.7**	6.3±1.5	6.6±1.4
	5	**4.4±0.7***	**5.1±1.1**	5.5±1.1	6.2±1.9	6.2±1.0	7.1±2.0
	10	**5.0±0.8**	**4.8±0.6***	**5.9±0.7**	6.6±1.2	6.8±1.0	7.3±1.3
	20	**6.0±0.5***	**7.2±1.8**	7.3±1.8	7.8±1.1	8.4±0.6	9.1±1.7
Vehicle	0	26.0±2.5	25.5±2.3	25.5±2.1	25.2±2.0	25.7±1.0	25.1±1.1
	5	30.1±2.4	28.2±2.2	27.6±2.0	27.4±1.3	27.2±1.6	26.5±1.5
	10	31.8±2.3	**28.4±2.2**	27.9±2.0	27.5±1.8	28.1±1.7	28.5±1.2
	20	32.3±2.6	29.9±2.7	28.7±2.5	29.0±2.2	29.5±2.0	29.8±1.7
Votes	0	**4.4±1.6**	**4.0±1.6***	**4.0±1.5***	**4.5±1.6**	4.7±1.9	5.0±1.5
	5	**4.4±1.4**	**4.3±1.5***	**4.4±1.8**	**4.5±1.5**	**5.1±1.7**	5.9±2.1
	10	**4.5±1.5***	**4.7±1.5**	**4.8±1.8**	**5.7±1.8**	**6.7±2.3**	7.3±2.0
	20	**4.8±1.7***	**5.8±1.9**	**7.8±2.9**	**9.5±2.9**	**11.2±3.1**	12.9±3.7
Waveform	0	**17.5±2.5***	**17.9±2.4**	**17.8±2.0**	18.8±1.4	19.0±1.0	20.1±1.2
	5	**17.0±2.6***	**17.3±2.0**	**17.7±1.9**	19.1±1.6	19.3±1.6	19.5±1.5
	10	**17.5±2.2***	**17.8±2.2**	19.5±1.6	20.8±1.7	21.2±1.7	21.9±1.8
	20	**18.1± 2.7***	**19.5±2.6**	19.3±2.0	22.0±1.5	22.2±1.8	22.8±1.7
Wine	0	7.6±4.5	**4.5± 2.5**	5.2±4.4	**4.4±2.4**	3.9±3.3*	5.1±3.1
	5	6.2±3.9	**3.4±2.4***	**5.2±2.9**	**4.9±3.0**	**4.7±2.9**	6.1±3.6
	10	**5.9±3.3**	**3.5±2.2***	**4.0±2.3**	**4.3±3.4**	5.8±4.2	7.3±4.1
	20	**5.6±3.4**	**4.2±2.4***	**5.9±3.9**	**7.0±3.1**	8.7±3.4	10.5±4.3

TABLE 3.9: Random forest test error I

Dataset	Noise (in %)	Bootstrap sampling ratio 10%	20%	40%	60%	80%	100%
Australian	0	13.3±1.3	6.5±0.6	**4.9±0.5**	**4.7±0.5***	**4.9±0.6**	5.1±0.8
	5	14.4±4.7	7.8±2.2	5.7±1.4	5.4±1.1	5.2±0.7*	5.4±0.8
	10	16.6±5.2	9.4±3.5	6.5±1.8	6.0±1.3	**5.7±1.0***	6.1±1.1
	20	21.5±6.5	13.5±4.8	9.6±2.5	8.9±2.0	8.7±2.0	8.3±1.5
Balance	0	16.9±2.0	15.4±1.8	14.5±1.9	**14.3±1.6**	**14.0±1.5***	15.0±1.8
	5	16.1±2.2	**15.2±2.0**	**14.6±2.0***	**14.8±1.8**	**15.4±1.9**	16.1±2.1
	10	**15.2±2.5**	**14.6±1.9***	**15.7±2.2**	**16.2±2.1**	17.1±2.2	17.6±2.4
	20	**15.2±2.2***	**16.0±2.1**	**17.5±2.6**	**19.1±2.4**	19.8±2.3	20.3±2.9
Breast W.	0	3.5±1.0	3.3±1.0	3.2±1.0	3.1±1.0	3.0±0.9*	3.0±1.0*
	5	3.4±1.1	3.3±0.9	3.2±1.0	3.2±1.0	3.2±0.9	3.1±1.0*
	10	**3.2±1.1***	**3.4±1.1**	3.7±1.3	3.8±1.2	3.8±1.1	3.8±1.1
	20	**3.7±1.2***	**4.2±1.4**	**5.0±1.5**	6.1±1.9	**5.8±1.5**	6.6±1.7
Diabetes	0	25.8±2.3	24.7±2.7	24.4±2.3	24.3±2.3	24.2±2.2	23.9±2.2*
	5	25.0±2.7	24.7±2.7	24.6±2.3	24.6±2.2	24.2±2.3*	24.4±2.2
	10	25.2±2.2	24.6±2.5*	24.7±2.3	25.1±2.1	25.0±2.3	24.7±2.1
	20	**25.4±2.5**	**25.3±2.5***	**26.5±2.5**	27.2±3.0	27.0±2.7	27.4±2.9
German	0	29.6±0.4	28.4±0.7	27.0±1.0	25.8±1.3	25.3±1.4	24.9±1.3*
	5	29.2±0.7	27.9±1.0	26.7±1.1	26.0±1.2	25.3±1.4	24.9±1.5*
	10	28.6±0.9	27.5±1.3	25.8±1.3	25.6±1.6	25.5±1.7*	25.5±1.5*
	20	28.0±1.3	27.2±1.6	26.6±1.6	26.4±1.6*	27.0±2.2	26.8±1.9
Heart	0	20.9±3.4	19.6±3.9	17.5±3.1	17.4±3.4	17.2±3.5*	17.5±3.2
	5	19.8±3.1	18.2±3.3	17.6±3.3*	18.7±3.7	18.4±3.3	17.7±3.4
	10	19.5±3.7	18.8±3.7	18.5±4.2*	19.1±3.4	19.8±3.7	18.9±3.8
	20	**19.7±4.3***	**20.3±4.7**	**21.5±3.5**	22.0±4.2	22.4±3.9	22.8±4.9
Hepatitis	0	20.5±1.1	17.9±2.6	14.9±3.0	13.7±3.4	13.1±3.3	12.7±3.6*
	5	19.6±2.2	15.7±3.1	13.9±3.3	13.0±3.3	13.0±3.7	12.5±3.7*
	10	17.7±3.5	15.7±3.7	13.9±3.8	13.9±3.6	13.2±3.4*	13.5±3.9
	20	16.3±4.2	15.5±4.0	15.8±4.2	**15.2±4.4***	16.1±4.4	16.5±3.8
Horse-Colic	0	30.2±1.7	27.6±1.6	26.5±1.8	26.5±1.9	26.2±1.8	25.3±1.8*
	5	31.0±3.1	27.6±2.7	26.3±2.2	25.8±3.0	25.8±2.9	24.8±2.9*
	10	31.2±3.4	28.3±3.6	27.1±3.0	25.7±3.6	25.8±3.3	25.6±3.3*
	20	31.2±4.1	29.8±3.8	28.1±3.6	27.4±4.3	27.2±3.9	26.5±3.7
Ionosphere	0	12.6±2.4	7.8±1.9	6.6±2.0	6.8±1.9	6.2±1.9	6.1±1.8*
	5	10.3±3.0	7.8±2.3	7.2±2.2	7.2±2.3	6.8±2.0*	7.1±2.3
	10	10.7±2.9	8.1±2.2	**7.4±2.3***	**7.5±2.3**	7.6±2.2	8.3±2.7
	20	11.1±3.1	**9.5±2.4***	**9.6±3.1**	**9.7±2.6**	10.8±3.2	10.6±2.7
Iris	0	4.4±2.5*	4.6±2.2	4.4±1.9*	4.7±2.5	5.0±2.6	4.8±2.4
	5	6.2±4.9	4.9±3.1*	5.5±2.8	5.0±2.7	5.3±2.9	5.4±2.9
	10	7.6±5.5	6.8±4.5	5.6±3.5*	5.7±2.8	5.7±3.0	6.5±3.9
	20	**8.2±4.8**	**7.8±5.0***	**8.9±5.0**	**8.9±5.3**	10.6±5.1	11.8±5.4

TABLE 3.10: Random forest test error II

Dataset	Noise (in %)	Bootstrap sampling ratio 10%	20%	40%	60%	80%	100%
Labor	0	12.0±3.7	12.7±3.2	11.7±2.8	11.1±2.9	9.0±2.8	8.9±2.2 *
	5	12.7±3.8	12.5± 3.3	13.5±3.6	12.5±4.8	12.4±3.8	11.0±3.3 *
	10	**12.7±4.2***	**12.8±4.5**	14.5±4.1	14.8±4.0	15.6±4.5	15.7±4.2
	20	**13.0±5.8***	**13.5±5.3**	15.5±5.1	15.7±4.8	16.4±4.7	16.4±4.5
Liver	0	36.8±2.0	33.5±2.2	29.7±3.0	28.1±2.9	27.5±3.2	27.1±3.2*
	5	35.1±3.2	32.7±3.0	29.9±3.2	29.2±3.5	28.8±3.6	28.5±4.0*
	10	33.6±2.9	31.4±3.9	30.8±4.2	30.4±3.6	30.3±3.7*	30.6±3.4
	20	33.9±3.8	33.2±4.2*	33.7±4.4	34.3±4.7	33.5±4.4	34.4±4.6
Lung Cancer	0	57.9±9.1	53.8±11.7	48.2±12.9	**43.0±13.1***	46.8±13.2	48.4±14.6
	5	60.7±7.4	55.8±11.7	49.2±13.0	49.3±12.3	47.6±13.7	47.4±12.6*
	10	61.8±9.3	55.9±11.7	54.2±12.9	50.2±15.7*	51.2±13.5	51.8±12.7
	20	61.8±10.9	58.7±12.1	55.4±11.5	54.7±13.0	**50.5±13.3***	54.8±14.2
Magic	0	14.4±0.4	13.6±0.4	12.9±0.3	12.6±0.4	12.4±0.3*	12.4±0.4*
	5	13.5±0.4	13.2±0.4	12.8±0.4	12.7±0.4	12.5±0.3*	12.5±0.3*
	10	13.3±0.4	13.0±0.4	12.9±0.4	12.8±0.4	12.7±0.4*	12.7±0.4*
	20	**13.6±0.4**	**13.5±0.4***	**13.5±0.4***	**13.6±0.4**	13.7±0.4	13.8±0.4
New-thyroid	0	8.4±2.3	7.3±2.5	5.1±2.0	3.3±1.8	3.0±1.0	4.4±1.2
	5	8.1±2.4	8.2±2.5	5.6±2.3	5.8±1.9	4.0±1.6	3.4±1.5
	10	8.2±2.8	5.9±2.1	3.3±2.4	4.8±1.8	4.3±1.7	3.2±1.7
	20	**6.1±2.5***	**6.2±3.0**	7.2±2.8	8.0±2.5	8.4±2.7	8.5±2.6
Ringnorm	0	13.2±1.4	6.5±0.7	4.9±0.5	**4.8±0.6***	**4.8±0.6***	5.0±0.7
	5	14.9±4.7	7.4±2.3	5.5±1.2	5.2±0.9*	5.2±0.9*	5.4±0.8
	10	16.7±5.2	9.3±3.1	6.2±1.4	6.1±1.2	6.0±1.3*	6.1±1.4
	20	21.4±5.7	14.3±4.8	9.7±2.4	8.6±2.3	8.5±1.7	8.2±1.8
Segment	0	5.9±0.9	4.4±0.9	3.5±0.5	2.9±0.6	2.7±0.7	2.6±0.6*
	5	5.9±0.8	4.5±0.9	3.7±0.7	3.1±0.8*	3.1±0.7*	3.2±0.8
	10	5.8±1.1	4.6±1.1	**3.4±0.6**	**3.0±0.7***	**3.4±0.7**	4.1±0.7
	20	5.7±0.7	**4.9±0.8**	**4.0±0.9***	**4.5±0.8**	**5.2±0.7**	6.1±0.7
Sonar	0	31.1±4.8	24.6±4.9	21.5±4.6	19.6±4.5	18.3±4.6*	18.6±4.4
	5	28.6±6.1	24.9±5.2	21.3±5.0	20.6±5.2	20.5±4.5	19.9±3.8*
	10	28.7±6.5	24.4±5.0	21.2±4.5	20.7±4.5	20.9±5.2	20.6±4.5*
	20	27.8±6.4	26.3±5.2	24.9±4.7	24.4±5.5	24.2±5.9	23.9±5.2*
Threenorm	0	18.2±0.9	16.9±0.9	16.0±0.9*	16.8±1.0	16.2±1.0	16.0±1.1*
	5	22.3±3.2	19.3±1.9	18.4±1.2	17.2±1.1	17.2±1.1	16.9±1.0*
	10	24.7±3.9	21.7±2.5	20.0±1.6	19.5±1.5	**18.6±1.6***	19.0±1.5
	20	30.6±4.5	26.3±3.2	23.4±2.0	22.9±2.2	22.3±2.0	21.6±1.7*
Tic-tac-toe	0	29.0±1.3	23.0±1.4	15.2±1.7	10.0±2.0	6.6±2.0	4.9±1.7*
	5	26.9±1.9	21.5±1.9	14.0±1.9	10.1±2.3	7.7±2.0	6.3±1.9*
	10	26.3±2.2	20.6±2.2	14.5±2.4	11.3±2.4	9.1±2.1	8.4±2.3*
	20	25.2±2.5	21.3±2.7	16.8±2.6	14.9±2.5	14.3±2.4	13.6±2.4*

TABLE 3.11: Random forest test error III

Dataset	Noise (in %)	Bootstrap sampling ratio 10%	20%	40%	60%	80%	100%
Twonorm	0	**3.3±0.3***	**3.3±0.3***	**3.3±0.3***	**3.4±0.3**	3.6±0.3	3.6±0.4
	5	4.5±1.3	3.9±0.8*	**3.9±0.5***	4.0±0.5	3.9±0.5*	4.0±0.4
	10	5.9±2.0	4.8±1.2	4.4±0.7*	4.6±0.9	4.4±0.6*	4.5±0.6
	20	9.5±4.4	7.4±2.7	6.3±1.4	6.0±1.2*	6.1±1.0	6.3±1.1
Vehicle	0	30.7±2.4	29.6±1.7	27.2±1.9	26.3±1.7	25.9±1.7*	26.1±1.6
	5	30.9±3.0	29.0±2.0	27.3±2.0	26.1±1.8	26.2±2.1	25.6±2.5*
	10	30.1±2.2	27.8±2.2	27.9±2.4	26.2±1.7	26.3±2.0	25.9±1.8*
	20	30.5±2.2	29.6±2.4	28.2±2.2	27.0±2.5	27.4±1.8	26.9±2.6*
Votes	0	5.3±1.7	4.4±1.5	3.9±1.4	3.6±1.3*	3.6±1.4*	3.6±1.3*
	5	5.4±1.7	4.4±1.5	4.0±1.4	3.9±1.4	3.7±1.5*	3.7±1.7*
	10	5.7±1.6	5.0±1.7	4.5±1.5	**4.1±1.5***	4.2±2.0	4.6±1.9
	20	6.3±2.2	**5.5±2.1***	**5.9±2.2**	6.2±2.3	6.5±2.6	6.7±2.5
Waveform	0	15.5±0.7	14.9±0.8	14.8±0.8	14.5±0.6*	14.6±0.6	14.6±0.6
	5	15.3±0.9	15.1±0.9	14.9±1.1	15.0±0.8	14.8±0.8*	14.8±0.6*
	10	15.1±0.6	14.8±0.5	14.8±0.8	14.9±0.8	14.6±0.9*	15.0±0.7
	20	14.9±1.0*	15.0±0.6	15.3±0.8	15.4±0.7	15.4±1.0	14.9±0.7*
Wine	0	3.0±1.8	3.1±1.9	2.5±1.7	2.3±1.6	2.1±1.7	1.9±1.6*
	5	4.8±3.3	3.6±2.7	2.7±2.0	2.9±2.2	2.8±2.1	2.4±2.0*
	10	5.8±4.3	4.1±3.0	3.7±2.6	3.4±2.6	3.2±2.4*	3.4±2.6
	20	7.0±4.4	5.8±3.6	5.9±3.5	5.1±3.1	5.3±3.5	5.0±3.0*

Chapter 4

A two-stage ensemble method for the detection of class-label noise

4.1 Abstract

The properties of bootstrap ensembles, such as bagging or random forest, are utilized to detect and handle label noise in classification problems. The first observation is that subsampling is a regularization mechanism that can be used to render bootstrap ensembles more robust to this type of noise. Furthermore, appropriate values of the sampling rate can be estimated using out-of-bag data. A second observation is that the ensemble classifiers tend to make more errors in incorrectly labeled instances. Thus, instances for which a sufficiently large fraction of ensemble predictors err are marked as noisy. Suitable values of this threshold, which are problem dependent, are determined by cross-validation using a wrapper method. Instances identified as noisy can then be either filtered (that is, discarded for training), or cleaned by correcting their class labels. Finally, an ensemble is built afresh on these cleansed training data. Extensive experiments in classification problems from different areas of application show that this procedure is effective to build accurate ensembles, even in the presence of high levels of class-label noise.

4.2 Introduction

The presence of noise, which is often unavoidable in real-world settings, is an important nuisance factor that needs to be taken into account in the design of learning algorithms [Frénay and Verleysen, 2014; Zhu and Wu, 2004b]. Two types of noise can be found

in the data used for automatic induction: class-label and feature noise [Zhu and Wu, 2004b]. Except when the latter is very strong, the presence of erroneous class labels is generally more harmful for learning and generalization [Zhu and Wu, 2004b]. According to its statistical properties label noise falls into three categories [Frénay and Verleysen, 2014]. In the *Noisy Completely at Random* (NCAR) model the probability of a mislabeled instance is uniform in feature and class spaces. In the *Noisy at Random* (NAR) model, the probability of mislabeled instances is independent of the feature values, but depends on the class. Finally, in the *Noisy Not at Random* (NNAR) models there are dependencies between the class-label noise, and both feature and class values. In this work, we assume that the noise is completely at random. Nevertheless, the method proposed can be readily adapted to take into account other types of class-label noise.

One way of alleviating the effects of noisy data is to incorporate some regularization mechanism into the design of the learning algorithm [Bi and Zhang, 2005; Freund, 2001; Wang and Li, 2017]. Another approach is to carry out a preprocessing step in which noisy instances are identified. These instances are then either discarded (filtering), or their class label corrected (cleaning) so that they do not interfere in the learning process. [Frénay and Verleysen, 2014]. In this work, we combine both strategies using randomized ensembles. Our goal is to take advantage of the robustness of bootstrap ensembles, such as bagging and random forest, to handle noise. Specifically, we will use subsampling as a regularization mechanism, which allows one to build ensembles that are robust to class-label noise [Sabzevari et al., 2014, 2015]. Another observation is that the individual classifiers in the ensemble tend to make more prediction errors on incorrectly labeled instances. Therefore, the fraction of incorrect ensemble predictions can be used as an indicator to detect noisy instances. Standard ensemble-based approaches to this problem remove instances for which more than half of the votes are incorrect (majority filtering) or where all votes are incorrect (consensus filtering). However, these choices need not be optimal. In this work, we propose to use a wrapper method to determine a near-optimal value for the percentage of incorrect votes necessary to identify a noisy instance.

The structure of the work is as follows: In Section 4.3, we present a review of previous work on learning from incorrectly labeled data. Particular emphasis is given to the use of subsampling to improve the robustness to class-label noise of bootstrap ensembles. The conclusions of this analysis are applied in Section 4.4 to the design of a method for noise detection. In Section 4.5, we present the results of an empirical evaluation of the proposed procedure and a comparison with other state-of-the-art noise detection methods. Finally, the conclusions of this work are summarized in Section 4.6.

4.3 Previous work

The problem of induction from noisy data has been extensively addressed in the area of ensemble learning [Bi and Zhang, 2005; Frénay and Verleysen, 2014; Freund, 2001; Sabzevari et al., 2014, 2015; Wang and Li, 2017]. Furthermore, as illustrated by recent implementations [García-Gil et al., 2017], these methods can be scaled to deal with class-label noise in large problems. Early on in the literature on ensembles, it was noted that the accuracy improvements of an ensemble with respect to a single learner are generally smaller when the training data are contaminated with class-label noise [Ali and Pazzani, 1994]. However, ensembles are generally more robust to this type of noise than single learners. In [Brodley and Friedl, 1999], an ensemble of three classifiers (a univariate decision tree, a k-nearest neighbor and a linear machine) is used to identify noisy instances. The noise detection protocol in this method is as follows: The training set is partitioned into 4-folds. Then, an ensemble is trained using three of these folds. The ensemble is then used to identify noisy instances in the fold not used for training. In this study, two filtering strategies are considered: majority filtering and consensus filtering. In majority filtering, a particular instance of the left-out fold is identified as noisy when the predictions of more than half of the learners are erroneous. The consensus filtering rule is more conservative. An instance is categorized as noisy when all the members of ensemble misclassify it. This process is repeated for each fold to identify noisy instances in all 4 folds. In this study, majority filtering exhibits better overall performance than consensus filtering. In [Khoshgoftaar et al., 2005] a similar approach is used, albeit with an ensemble of 25 classifiers of different types. The increased diversity of this heterogeneous ensemble reduces the risk of false positives in the noise detection process. In addition, different intermediate strategies between majority and consensus filtering are explored. In the problems investigated, the optimal threshold of erroneous ensemble predictions used to identify an instance as noisy was close to consensus filtering.

In [Verbaeten and Van Assche, 2003], an ensemble of Top-down Induction of Logical Decision Trees (Tilde) is used to filter the training data. In this study either cross-validation or bootstrap sampling are used to generate the training sets on which the base learners are built. Majority and consensus filtering are then applied to identify noisy instances. The best results are obtained with majority filtering. In addition, the use of boosting in noise detection problems is explored. The idea is to filter out instances whose weights are above a threshold after a predefined number of iterations of the boosting algorithm. In the problems investigated, this strategy does not perform well.

In [Zhu et al., 2003], a distributed method for large datasets is presented. In this method, the original training set is partitioned into small subsets. A classifier is induced from

each of the these subsets. These classifiers are then used to classify the instances in the complete training set. The local error of an instance is defined as the fraction of classifiers whose training data included that particular instance and misclassified it. The global error for an instance is computed using the classifiers that did not include that instance in their training sets. Majority and consensus filtering are used to identify noisy instances. A necessary condition for an instance to be identified as noisy is that it be misclassified by the base learners whose training sets induced it. The justification is that a classifier has usually higher accuracy on instances that are in its training set. In this work, majority filtering is found to yield better results than consensus filtering.

In [Zhong et al., 2005], a noise detection strategy that combines ideas from bagging and boosting is investigated. In this method, all training instances are initially assigned equal weights. Then, bootstrap samples from the original training data are used to build different classifiers. For each instance a noise count is computed as the number of base learners that have misclassified the instance. The noise count is used in a boosting-like iterative process in which the weight of the instances with higher noise counts is decreased. This process is iterated for a predefined number of rounds. Finally, instances whose noise count values are higher than a threshold are labelled as noisy. Ensemble methods based on ranking have been explored in [Sluban et al., 2014]. In this work, instances are ranked according to the number of base learners that make incorrect predictions. In the medical dataset analyzed, a domain expert singles out the instances with highest ranks for further analysis. Expert knowledge is then used to determine the type of anomaly of the instances with the highest ranks (labeling error, outlier, complex medical case, etc.).

In most of the noise detection methods based on ensembles, majority or consensus filtering are used. Majority filtering was found to be better than consensus filtering in most studies in which homogeneous ensembles were used [Brodley and Friedl, 1999; Sluban and Lavrač, 2015; Verbaeten and Van Assche, 2003; Zhu et al., 2003]. By contrast, in small heterogeneous ensembles, agreement rates close to consensus filtering seem to be more effective [Khoshgoftaar et al., 2005; Sluban and Lavrač, 2015]. A qualitative explanation of this observation can be given: In homogeneous ensembles the base learners are of the same type. Diversity is commonly obtained using randomization strategies, such as bootstrap sampling (e.g. in bagging and random forest). Thus, the decision boundaries of the individual classifiers are similar to each other. In fact, the decision boundary is a refinement of the decision border of the individual base learners [Martínez-Muñoz and Suárez, 2005]. Therefore, noisy examples close to this decision boundary can be detected only if majority voting is used. By contrast, in heterogeneous ensembles, the decision borders are more varied because the individual classifiers are of different types. For this reason the dispersion of class labels assignments may simply reflect this

variability. In consequence, higher agreement rates should be used to mark an instance as noise. In summary, the optimal agreement rates for noise detection depends on the classification problem and type of ensemble. However, as far as we are aware, the influence of the agreement rate on the effectiveness of the noise detection method has not been systematically evaluated hitherto.

4.4 A wrapper method for class-label noise detection

In this work, ensembles are used to detect and handle noise in the class labels of the training instances. We focus on randomized ensembles, such as bagging and random forest, in which the individual classifiers are trained on bootstrap samples of the original data. The design of the algorithm leverages two observations: The first one is that reducing the sampling rate in the bootstrapping step can make the ensemble more robust to class-label noise [Sabzevari et al., 2014, 2015]. A second observation is that the fraction of erroneous predictions by the ensemble classifiers necessary to detect noisy instances depends on the classification task at hand. Therefore, the value of the threshold θ used to mark an instance as noisy should be estimated for each problem during the training phase. The method proposed proceeds in two stages: First, the optimal sampling rate for the bootstrapping step in the construction of the ensemble is determined using out-of-bag data. Then, the optimal threshold for disagreement between the ensemble predictions and the actual class of the instance is determined by means of a wrapper method. Finally, instances labeled as noise are either removed (*filtering*) or relabeled (*cleaning*) and an ensemble is built afresh from the cleansed training data.

The pseudocode of the method is detailed in Algorithm 4. In the first stage of the algorithm (steps 1 to 8 in Alg. 4), ensembles are built using different values of the sampling rate: $0.1, 0.2, 0.4, 0.6, 0.8, 1.0$, and 1.2. From these, H_T^*, the ensemble that is expected to generalize best is selected and kept for the next phase. In our implementation, out-of-bag instances are used to estimate the generalization error and carry out this selection. In this way, an unbiased selection is achieved, independently of the amount of noise present in the dataset. According to empirical evidence [Sabzevari et al., 2014, 2015], the higher the level of class-label noise, the lower the value of the subsampling rate that is selected. Although the possibility that the presence of noisy instances lead to an incorrect selection of the best ensemble cannot be ruled out, we have not observed this effect in the experiments carried out. This is probably due to the fact that random forests are fairly robust to class-label noise, even without cleansing.

In the second stage (steps 9 to 14 in Alg. 4), the predictions of the base classifiers of the selected ensemble, H_T^*, are used to identify noisy instances. Different values of the

Algorithm 4: Noise detection using bootstrap ensembles

Input: $\mathcal{D}_{train} = \{(\mathbf{x}_n, y_n)\}_{n=1}^{N_{train}}$ % Training set
bootstrap_ensemble % Bootstrap ensemble method
T % Ensemble size
wrapper_learner % Wrapper learner method
cleanse_type % either filtering or cleaning
Output: $\mathcal{D}_{cleansed}$ % Cleansed set

1 $min\_error \leftarrow \infty$ % determine optimal sampling rate
2 **foreach** *sampling_rate* **in** $[0.1, 0.2, 0.4, 0.6, 0.8, 1.0, 1.2]$ **do**
3 $H_T \leftarrow bootstrap\_ensemble(\mathcal{D}_{train}, sampling\_rate, T)$
4 $error \leftarrow estimate\_error(H_T, \mathcal{D}_{train})$
5 **if** $error < min\_error$ **then**
6 $min\_error \leftarrow error$
7 $sampling\_rate^* \leftarrow sampling\_rate$
8 $H_T^* \leftarrow H_T$
9 $min\_error \leftarrow \infty$ % determine optimal disagreement rate
10 **foreach** θ **in** $[0.5, 0.6, 0.7, 0.8, 0.9, 1.0]$ **do**
11 $\mathcal{D}_{cleansed} \leftarrow cleanse\_with\_oob(\mathcal{D}_{train}, cleanse\_type, H_T^*, \theta)$
12 $error \leftarrow cv\_error(wrapper\_learner, \mathcal{D}_{cleansed}, K = 3)$
13 **if** $error < min\_error$ **then**
14 $\theta^* \leftarrow \theta$
15 $\mathcal{D}_{cleansed} \leftarrow cleanse\_with\_oob(\mathcal{D}_{train}, cleanse\_type, H_T^*, \theta^*)$

threshold θ for the disagreement rate are considered; namely, $0.5, 0.6, 0.7, 0.8, 0.9$, and 1.0. For a given value of θ, instances for which the fraction of incorrect predictions given by the classifiers in the selected ensemble is above this threshold are marked as noise (step 11). Since the data being cleaned and the data used to train H_T^* are the same, only the predictions of base classifiers in H_T^* whose training set does not include that particular instance are used (i.e. out-of-bag instances). Cleansing consists in either correcting the label of noisy instances (*cleaning*), or eliminating them (*filtering*). Then, we use a wrapper method and compute estimates of the generalization error of a learner built on the cleansed training data. In our implementation, this error is estimated using K-fold cross-validation. The optimal value of the disagreement threshold, θ^*, is the one that minimizes this estimate of the generalization error. The details of this selection are as follows: In each of the K iterations of the cross validation procedure, one of the K folds is set apart for validation. Then, a classifier is trained on remaining $K-1$ folds cleansed using the corresponding value of θ. The classifier is then evaluated on the left-out fold, which is not cleansed. Finally, the cross validation error is calculated by averaging the errors of the validation folds in each of the K iterations. The value selected, θ^*, is the threshold that minimizes this cross-validation estimate of the generalization error. This optimal threshold value, θ^*, and the ensemble H_T^* are then used to clean or filter the

training data. Note that the type of classifier used in the wrapper step is a parameter of the algorithm. In general, we think it is preferable to use the same classifier as the one that is eventually trained with the cleansed data. In the following section we carry out an extensive empirical analysis of the accuracy and resilience to noise of ensembles built in this manner.

4.5 Empirical evaluation

To assess the effectiveness of the proposed noise detection procedure extensive experiments have been carried out in 19 binary classification problems from the UCI repository [Bache and Lichman, 2013]. The characteristics of the datasets are summarized in Table 4.1. The implementation makes use of the R *randomForest* package [Liaw and Wiener, 2002a] and the R *adabag* package for AdaBoost [Freund and Schapire, 1996]. In both packages the default settings are used. Specifically, for random forests, the number of variables considered for splitting at the inner nodes of the random trees is the square root of the total number of attributes in the problem. The minimum size of a terminal node is set to 1. Trees in the forest are grown to their maximum possible size. For AdaBoost, weighted resampling is used. The coefficient that controls the weight update is $\alpha = 1/2ln((1 - err)/err)$.

In all the classification tasks, with the exception of *Ringnorm*, *Threenorm* and *Twonorm*, which are synthetic problems, the labeled instances are randomly assigned to the training and test sets. The sizes of these sets are 2/3 and 1/3 of the original set, respectively. For synthetic problems, 300 examples are used for training and 2000 for testing. In all cases, stratified sampling is used. The results reported are averages over 50 executions. In these executions, different random partitions of the data into training and test sets, or, in synthetic problems, independent realizations of the data are used. The following protocol is followed for each classification task and execution:

1. First, noise is injected in the data by randomly switching the class label of a random subset of the training instances. Different noise rates are considered: 0% (no noise is injected), 10%, 20%, 30% and 40%. This type of class-label noise is known as completely at random noise (NCAR) [Frénay and Verleysen, 2014].

2. For each noise level, ensembles composed of 501 trees are trained using the following bootstrap sampling ratios: 10%, 20%, 40%, 60%, 80%, 100% (standard) and 120%. Then, the out-of-bag error is computed for each ensemble. The ensemble with the best out-of-bag accuracy (H_T^*) is kept for the next step. Both random

Dataset	**Training**	**Test**	**Attrib.**
Australian	460	230	14
Blood transfusion	499	499	5
Boston	337	169	14
Breast	466	233	9
Chess	2130	1065	37
Crx	460	230	15
Diabetes	512	256	8
German	667	333	20
Heart	178	92	13
Horse-Colic	246	122	21
Ionosphere	234	117	34
Liver	230	115	6
Parkinsons	130	65	22
Ringnorm	300	2000	20
Sonar	491	208	60
Spambase	3067	1534	58
Threenorm	300	2000	20
Tic-Tac-Toe	639	319	9
Twonorm	300	2000	20

TABLE 4.1: Characteristics of the classification problems used in the empirical evaluation

forests and bagging ensembles of unpruned CART trees have been considered. According to the results of our experiments, they are both equally effective to identify noisy instances. Therefore, since random forest are generally more accurate than bagging ensembles, only results for the former are reported.

3. For each instance in the training set, we compute a tally of votes (class label predictions). In this tally, only the predictions of those classifiers in H_T^* whose training sets do not include that particular instance are used.

4. Instances are tentatively marked as noisy if the percentage of incorrect predictions by the individual ensemble classifiers is above a specified threshold θ. Noisy instances are either cleaned (i.e. their class labels are corrected by assigning the majority label in the out-of-bag predictions) or filtered (i.e. removed from the training set). The following values of the threshold θ are tested: 0.5 (majority filtering), 0.6, 0.7, 0.8, 0.9 and 1.0 (consensus filtering). As described in the previous section, the optimum value θ^* is selected by K-fold cross-validation within the training set by means of a wrapper method. From the results of exploratory experiments with different values of K, reliable estimates are obtained with $K = 3$, which is the value used in the experiments. Random forest composed of 501 trees is used in this wrapper stage.

5. The training instances for which the fraction of incorrect out-of-bag predictions is above θ^* are definitively marked as noise. These noisy instances are then either corrected or removed from the training set.

6. Finally, a random forests of 501 random trees, is built on the cleansed training set. The labels *Fl_rf* for random forest with filtering and *Cl_rf* for random forest with cleaning will be used throughout the experiments.

7. The generalization error of the resulting ensembles is estimated on the unperturbed test set.

As a benchmark, a second *cleansed* dataset is obtained using majority filtering following the proposals of previous studies [Brodley and Friedl, 1999; Verbaeten and Van Assche, 2003; Zhu et al., 2003], and, in particular, for homogeneous ensembles [Sluban and Lavrač, 2015]. In this benchmark, the data are filtered using K-fold cross-validation. At each iteration, an ensemble is trained using data from K-1 folds. Then, the ensemble is used to predict the labels of the instances in the remaining fold. The examples of the holdout fold that are incorrectly classified are removed from the dataset. The process is repeated to filter the other folds. We implemented this method using random forest of size 501 and $K = 3$. Finally, a random forest composed of 501 random trees is built on the training set cleaned with majority filtering. The label *Fl_maj_rf* is used to denote this benchmark. As an additional reference, we also present the results of standard random forest (labelled as RF) and AdaBoost (labeled as Boosting) trained on the original uncleaned training set.

4.5.1 Predictive accuracy

The results of the experiments carried out to compare the accuracies of the different methods considered are summarized in Tables 4.2, 4.3 and 4.4. The test errors reported correspond to averages over 50 realizations of the training and test sets. These are averages followed by their standard deviations after the $\pm$ sign. To assess the significance of these observations, the results of an overall comparison of the different methods are summarized in Fig. 4.1 using the methodology introduced in [Demsar, 2006]. In these plots, the average ranks of the different methods are compared. The differences between the average ranks of two methods are statistically significant at a level $\alpha = 0.05$ if they are above a critical distance (CD). Methods whose average ranks are not significantly different are linked by a horizontal line. Average rank plots are presented for experiments with different levels of class-label noise injected: 0%, 10%, 20%, 30% and 40%.

TABLE 4.2: Test errors of the different methods for different noise levels (I)

Dataset	Noise (in %)	*Fl_rf*	*Cl_rf*	*Fl_maj_rf*	Boosting	RF
Australian	0	13.3±1.9	13.5±1.9	13.8±2.0	**13.1±1.9**	**13.1±1.9***
	10	13.6±1.9	13.7±1.9	13.7±2.0	18.2±2.2	**13.5±1.9**
	20	14.1±1.8	14.0±2.0	**13.8±2.1**	23.8±3.1	15.3±2.2
	30	16.3±2.9	16.1±2.8	**15.8±2.6**	33.6±3.6	21.7±3.0
	40	26.0±6.3	**24.5±5.2***	26.1±5.1	41.9±3.2	34.3±3.5
Blood transfusion	0	21.9±1.6	**21.8±1.9**	22.1±1.8	25.7±2.3	24.7±2.1
	10	22.2±1.5	**22.0±1.7**	22.9±2.0	28.0±2.8	26.4±2.1
	20	23.3±2.3	**23.2±2.4**	24.9±2.6	31.2±3.2	29.8±2.8
	40	34.0±4.6	**32.6±4.9***	38.5±4.5	43.2±4.3	42.1±4.5
Boston	0	13.2±2.2	13.5±2.2	14.7±2.7	**12.4±2.4**	12.8±2.1
	10	**13.7±2.6**	13.8±2.6	14.6±2.2	16.3±2.8	**13.7±2.6**
	20	15.5±2.8	15.5±2.4	**15.2±2.8**	23.6±3.9	17.8±2.7
	30	19.2±3.8	**18.9±3.7**	20.2±4.0	32.5±4.2	25.1±4.4
	40	26.8±6.1	**26.1±6.5**	28.8±6.5	41.1±4.1	36.2±5.1
Breast	0	**3.1±1.1***	**3.1±1.0***	3.3±1.0	3.4±1.1	**3.1±1.1***
	10	3.6±1.2	3.6±1.1	**3.5±1.2**	8.7±1.8	4.1±1.3
	20	**4.3±1.5**	4.4±1.8	**4.3±1.4**	14.9±2.5	6.7±1.9
	30	**6.6±3.4***	**6.6±3.2***	7.6±3.5	24.1±3.8	12.7±3.8
	40	15.5±6.4	**14.8±5.8**	18.4±4.8	33.8±4.2	26.4±4.7
Chess	0	1.7±0.4	1.7±0.4	2.6±0.4	**0.4±0.2***	1.6±0.4
	10	**2.2±0.4***	2.3±0.4	3.0±0.6	4.1±0.8	2.3±0.5
	20	**3.4±0.7***	3.6±0.8	3.7±0.8	5.8±0.8	4.1±0.6
	30	**5.5±1.2**	6.0±1.1	5.7±1.0	11.0±1.7	10.0±1.1
	40	**16.6±3.0***	17.7±3.2	17.7±2.3	24.6±2.2	25.7±1.7
Crx	0	12.9±1.8	13.0±2.1	13.2±2.1	13.8±1.8	**12.6±1.9**
	10	13.7±1.8	**13.6±1.8**	**13.6±2.0**	18.8±2.5	14.0±1.9
	20	14.1±2.2	14.2±2.3	**14.0±2.2**	25.3±3.1	16.2±2.1
	30	16.9±3.3	**16.2±2.9**	17.0±3.0	33.4±3.8	22.3±3.3
	40	24.5±5.9	**24.3±5.9**	25.8±5.0	41.5±3.4	33.8±3.7
Diabetes	0	24.1±2.0	24.3±1.8	24.3±1.9	27.5±2.1	**24.0±1.9**
	10	24.6±2.3	24.5±2.3	**24.3±2.2**	30.1±2.9	25.4±2.2
	20	25.4±2.5	25.4±2.4	**25.3±2.4**	34.4±3.0	27.8±2.5
	30	27.3±2.6	**26.7±2.7***	27.3±2.7	39.6±3.3	31.7±3.0
	40	**32.5±4.9**	**32.5±3.8**	33.4±4.7	44.4±4.0	39.3±4.5

TABLE 4.3: Test errors of the different methods for different noise levels (II)

Dataset	Noise (in %)	*Fl_rf*	*Cl_rf*	*Fl_maj_rf*	Boosting	RF
German	0	24.5±2.2	24.6±1.9	26.5±2.1	25.1±2.1	**24.2±2.0**
	10	25.0±1.9	25.6±2.4	26.7±2.4	28.1±2.2	**24.8±1.9**
	20	**26.5±2.7**	26.7±2.3	27.1±2.2	32.7±3.1	27.1±2.7
	30	28.9±2.7	28.6±2.8	**28.3±2.6**	38.3±3.0	31.0±2.7
	40	32.2±4.2	**31.7±3.9**	32.8±4.0	43.3±2.9	37.9±3.6
Ionosphere	0	6.9±2.0	6.9±2.0	7.3±1.9	**6.4±1.8**	6.8±1.9
	10	**7.7±2.0**	7.8±2.2	8.0±2.4	11.3±3.3	7.8±2.1
	20	9.4±3.1	9.5±3.7	**9.3±3.0**	19.0±4.2	11.3±3.7
	30	14.8±4.5	14.1±4.0	**13.8±4.6**	28.0±5.3	18.4±3.8
	40	32.2±4.2	**31.7±3.9**	32.8±4.0	43.3±2.9	37.9±3.6
Heart	0	**17.5±3.4**	18.1±3.8	17.9±3.8	20.9±3.4	18.1±3.6
	10	19.6±4.3	19.7±4.3	**18.9±4.8**	26.3±4.1	20.3±3.8
	20	21.7±4.0	**21.4±4.5**	21.5±4.2	32.1±5.8	23.5±4.3
	30	24.2±5.6	24.2±5.5	**24.0±5.3**	37.1±5.3	28.7±5.7
	40	34.2±7.6	**33.7±7.2**	34.5±7.3	43.2±6.6	38.4±6.2
Horse-Colic	0	15.6±2.5	**15.5±2.7**	17.6±3.6	15.7±2.6	**15.5±2.7**
	10	**17.5±3.0**	**17.5±3.1**	18.2±3.4	22.3±3.4	**17.5±2.9**
	20	**20.2±3.6**	20.5±4.1	20.3±4.0	28.3±4.9	20.7±3.7
	30	26.2±4.7	26.5±4.9	**26.0±5.5**	35.4±4.5	29.0±4.9
	40	36.0±6.3	**35.2±6.5**	36.0±6.5	43.6±4.7	39.7±4.5
Liver	0	28.4±4.2	29.7±3.8	31.2±4.4	29.6±3.4	**27.4±3.8***
	10	**31.4±4.3***	32.7±4.9	33.6±4.8	34.4±3.9	**31.4±4.4***
	20	**34.5±4.6**	35.5±4.2	36.0±4.7	37.7±4.6	34.6±4.5
	30	38.0±5.9	38.5±5.3	38.8±5.7	40.5±5.5	**37.7±5.8**
	40	43.8±5.6	**43.1±5.4**	44.6±4.6	44.9±4.6	43.5±5.3
Parkinsons	0	11.9±3.5	11.4±3.8	16.3±4.0	**8.1±3.5***	11.0±3.5
	10	13.5±3.8	13.5±3.8	16.4±4.0	**12.8±3.9**	13.0±3.7
	20	**17.8±4.7**	18.1±5.2	18.1±4.5	22.0±5.0	**17.8±4.7**
	30	21.0±5.2	21.5±4.7	**20.7±5.4**	29.8±7.1	23.9±5.5
	40	31.8±8.2	31.8±8.0	**29.3±7.5***	39.0±6.2	34.6±7.1
Ringnorm	0	6.1±1.0	6.2±1.0	8.3±1.4	**4.4±0.4***	6.0±1.0
	10	6.8±1.3	6.9±1.1	8.7±1.5	8.6±1.1	**6.7±1.1**
	20	8.4±1.8	**8.3±1.6**	9.9±2.2	15.2±1.7	**8.3±1.7**
	30	11.8±2.7	**11.5±3.1**	12.2±3.2	24.4±2.7	12.8±2.4
	40	21.1±5.3	**18.2±5.3***	20.8±6.4	36.3±3.1	24.7±3.5

TABLE 4.4: Test errors of the different methods for different noise levels (III)

Dataset	Noise (in %)	*Fl_rf*	*Cl_rf*	*Fl_maj_rf*	Boosting	RF
Sonar	0	18.8±3.9	19.4±4.8	24.5±3.8	**14.4±3.8***	18.5±3.7
	10	20.7±5.2	21.5±4.9	25.2±4.6	20.1±4.4	**19.9±4.7**
	20	24.2±4.8	24.3±5.4	26.1±5.2	26.9±5.5	**23.7±5.5**
	30	31.7±7.1	31.9±7.2	33.1±7.7	35.1±6.6	**31.2±7.6**
	40	**38.6±6.2**	39.7±7.5	39.6±7.6	42.6±7.7	39.4±7.4
Spambase	0	5.1±0.6	5.1±0.6	6.3±0.5	**4.9±0.4**	5.0±0.5
	10	**5.9±0.5***	6.1±0.7	6.5±0.5	9.4±0.8	6.6±0.6
	20	**6.7±0.6**	**6.7±0.6**	6.9±0.7	13.7±1.1	9.3±0.9
	30	**7.8±0.8***	**7.8±0.8***	8.6±0.9	20.3±1.4	14.6±1.1
	40	11.8±2.2	**11.1±1.5***	14.3±1.5	32.0±1.6	25.6±1.3
Threenorm	0	17.0±1.0	17.2±1.3	19.2±1.4	16.8±0.9	**16.5±0.9**
	10	18.8±1.5	19.1±1.6	20.5±1.8	20.7±1.4	**18.4±1.2***
	20	20.9±2.2	21.2±2.6	21.7±2.5	25.5±1.6	**20.8±1.8**
	30	**25.6±2.6**	25.7±2.5	26.7±2.5	32.3±2.1	26.5±2.1
	40	33.4±4.5	**33.0±4.7**	34.3±5.1	40.6±2.9	35.7±3.1
Tictactoe	0	2.5±0.9	2.4±1.1	7.2±2.5	**0.6±0.5***	2.3±1.0
	10	**5.7±2.2**	6.2±2.3	12.0±2.6	10.4±1.7	5.8±1.6
	20	12.5±2.9	13.8±2.8	17.4±2.8	21.3±2.3	**12.4±2.3**
	30	22.1±2.7	23.5±2.6	24.1±3.3	30.5±2.6	**21.7±2.2**
	40	**34.4±3.7**	**34.4±3.5**	35.1±3.5	40.5±3.7	35.7±3.2
Twonorm	0	3.9±0.5	3.9±0.5	4.3±0.5	**3.7±0.3**	3.8±0.4
	10	4.6±0.7	**4.5±0.6**	**4.5±0.5**	7.7±1.0	4.9±0.7
	20	5.6±1.2	5.5±0.9	**5.2±0.9**	13.7±1.6	6.5±1.0
	30	8.2±2.9	7.7±2.8	**7.3±2.3**	23.7±2.4	10.8±1.8
	40	17.5±6.3	16.7±6.9	**15.2±5.5**	35.7±3.2	23.2±3.3

From the results presented in Fig. 4.1, Tables 4.2, 4.3, and 4.4, one concludes that random forests with optimal filtering or cleaning (*Fl_rf* and *Cl_rf*) are among the most accurate ensembles at all noise levels. When no noise is injected, boosting and random forest trained on the original data are slightly better than *Fl_rf* and *Cl_rf*. However, the differences are not statistically significant. Because of its progressive emphasis on incorrectly classified instances, boosting is not robust to errors in the class labels. This is apparent from the degradation of its performance in problems with higher levels of noise. In fact, boosting has the worst average rank for 10% to 40% noise levels. In these cases, the differences with most other methods are statistically significant. The differences between filtering and cleaning are not statistically significant. Filtering seems to have a slightly better performance at lower noise levels. When either 30% or 40% of the class labels of the training instances are perturbed, cleaning is slightly better than filtering. This is probably a consequence of the loss of information that results form discarding

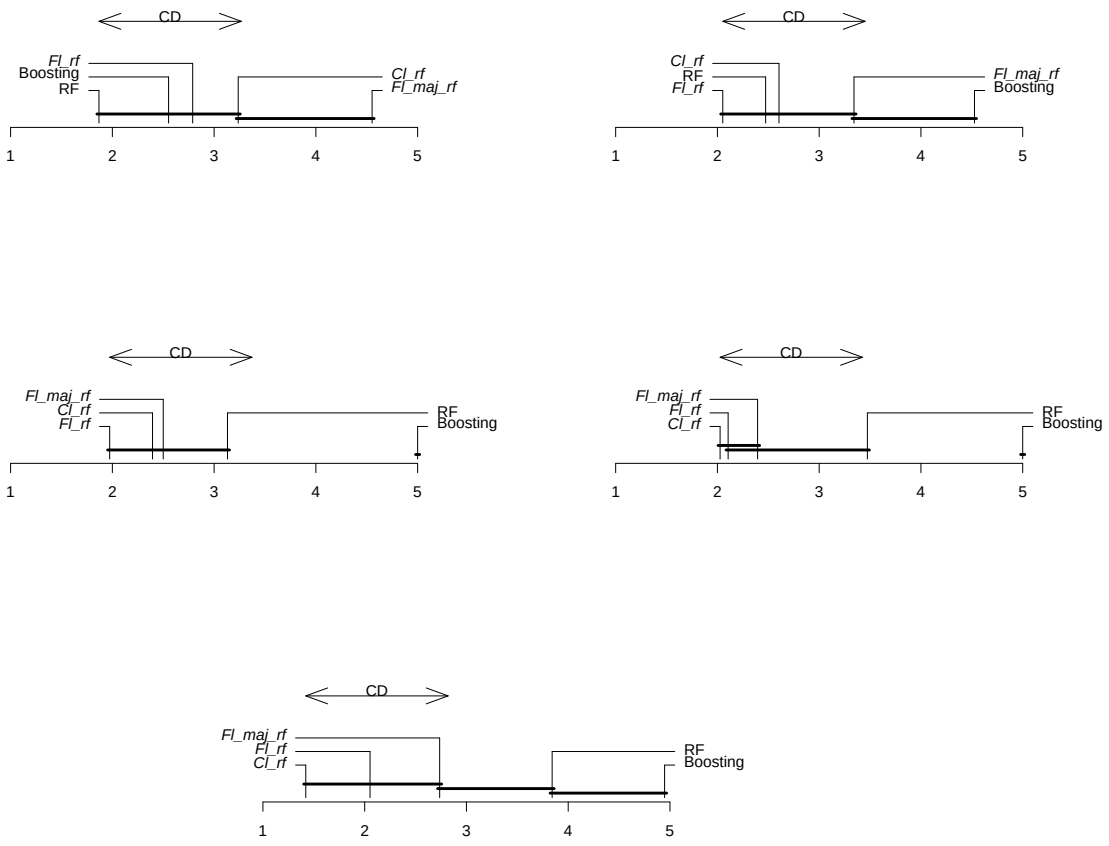


FIGURE 4.1: Comparison of the average ranks of the different types of ensembles for different levels of class-label noise: without injected noise (top left); 10% (top right); 20% (middle left) ; 30% (middle right); 40% noise (bottom). Horizontal lines connect methods whose average ranks are not significantly different according to a Wilcoxon signed-ranks test (p-value < 0.05).

instances in filtering. Finally, we observe that selecting adequate values of the threshold θ for noise filtering or cleaning is superior to using standard majority filtering at all noise levels. These improvements are more pronounced at lower noise levels.

4.5.2 Optimal values of the hyperparameters

In this section, we analyze the trends in the values selected for the sampling rate and the threshold, θ^*. We consider first the values of the sampling rate. In the top plot of Fig. 4.2 the average sampling rates obtained by the proposed cleaning procedure for each dataset and noise level are displayed. One can observe that in the original unperturbed problems (white bars in the plot) the optimal values for this hyperparameter are strongly

problem dependent. For instance, in *Chess*, *Spambase*, and *Tic-tac-toe* these values are above the standard 100% resampling rate. It is likely that for these problems the variability random forests is too large, and that oversampling is an effective mechanism to reduce it. In the remaining problems subsampling, which increases the variability of the ensemble, seems to be more effective. In some cases, such as *Blood Transfusion*, *Breast*, *Ionosphere*, and *Twonorm*, the optimal values of the sampling ratios are fairly low (around or below 30%, on average). In general, as more class-label noise is injected in the data, the optimal sampling ratios become smaller. This confirms the observation that subsampling becomes more effective as the amount of class-label noise increases [Sabzevari et al., 2014, 2015].

In the method proposed, an instance is marked as noisy when it is incorrectly labeled by a fraction of classifiers that exceeds a specified threshold θ^*. The optimal value of this parameter is also strongly problem-dependent. However, from the results presented in the bottom plot of Fig. 4.2, it is apparent that as more noise is injected, the values of θ^* become smaller and approach 0.5, which corresponds to majority voting. This is consistent with the observation that majority filtering becomes more effective at higher noise levels.

4.5.3 Noise detection

The value of the threshold used to mark an instance as noisy is determined on the basis of the accuracy of the wrapper classifier trained on cleansed data. The question remains whether this procedure is actually effective for the detection of noisy instances. To investigate this issue, we have recorded the fraction of instances marked as noise for the different classification tasks and with different levels of injected noise. The results of these experiments are presented in the plots of Fig. 4.3 for the following noise levels: 0% (first row), 10% (second row) and 30% (third row). Each plot of this figure shows the average percentage of instances that are marked as noise with a white bar. From those instances, the ones corresponding to the artificially injected noise are marked in red. The results for the proposed cleaning procedure using filtering are summarized in the plots in the left column of this figure. For reference, the results of majority filtering benchmark are displayed in the plots in the right column.

An analysis of the results for the unperturbed classification tasks (first row of Fig. 4.3) reveals that the levels of detected noise vary significantly. In the synthetic problems, which, by construction, are noiseless, the proposed cleaning procedure using filtering discards only a small percentage instances. By contrast, in some problems (such as *Blood transfusion*, *Diabetes*, *German* and *Liver*) a significant fraction of instances are

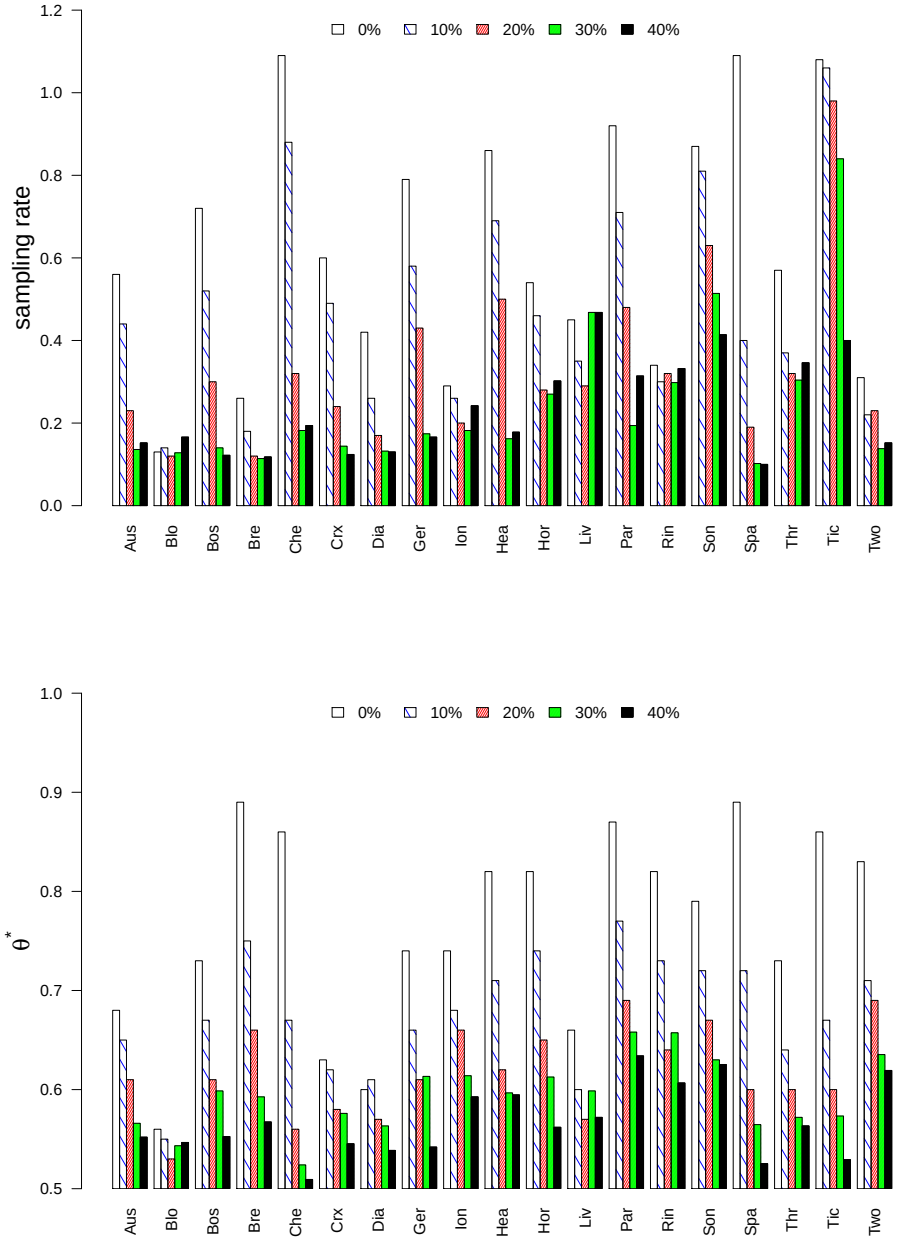


FIGURE 4.2: Optimal sampling rate (top) and threshold for cleansing (bottom) for random forest with filtering (*Fl_rf*)

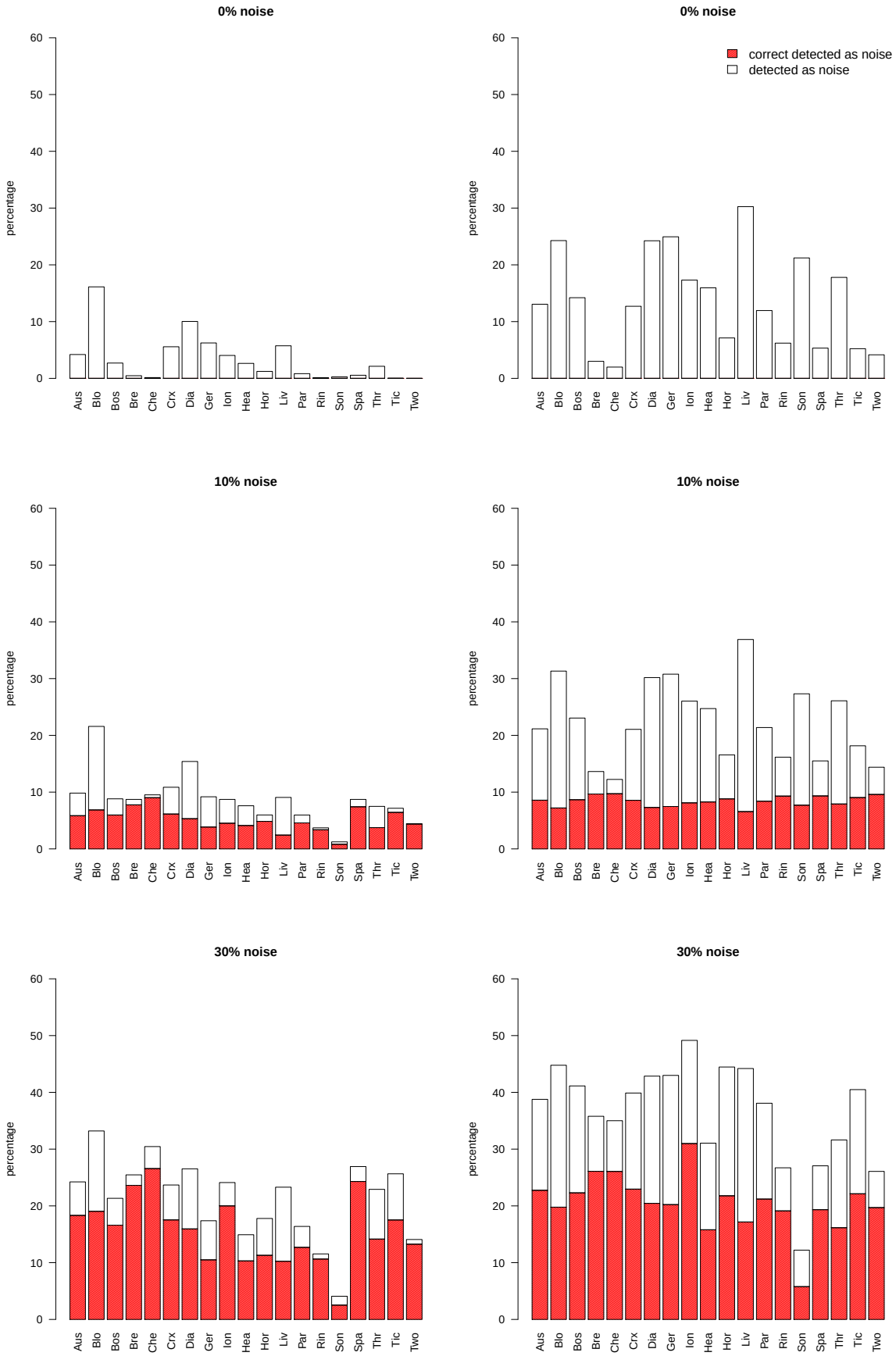


FIGURE 4.3: Percentage of filtered examples (white bars) and filtered examples that correspond injected noise (red part of the bars) for different noise levels: without injected noise (first row), 10% (second row), and 30% (third row) for the proposed cleansing procedure (left column) and majority filtering (right column)

identified as noise. Filtering these instances does not appear to be detrimental. As a matter of fact, in these problems the proposed cleaning procedure yields competitive or better accuracy rates with respect to random forest trained on the uncleaned data (see Tables 4.2 and 4.3).

For the unperturbed classification tasks, majority filtering is much more aggressive and marks many more instances as noise than the proposed procedure. In problems without noise in the class labels, such as *Threenorn*, *Tic-tac-toe*, *Ringnorm*, *Twonorm*, around or above 5% of the training instances are discarded (see upper right plot on Fig. 4.3). As a result, there is a significant decrease of the accuracy for random forest trained on these cleansed data (see Tables 4.3 and 4.4). In real-world problems, it is not possible to know the level of intrinsic noise. Nonetheless, majority filtering is likely to discard too many instances as well. Specifically, this method marks more than 20% of the instances as noise in five of the datasets analyzed (*Blood transfusion, Diabetes, German, Liver* and *Sonar*). The accuracy of random forest trained on the data cleansed by majority filtering in *Blood transfusion* and in *Diabetes* is fairly good. However, in *Liver* and *Sonar* it is the least accurate among the methods considered.

The results of experiments on classification tasks perturbed with 10% and 30% class-label noise, are displayed in the second and the third rows of Fig. 4.3, respectively. The red part of the bars is the percentage of instances whose class-label has been switched in the noise injection process that are identified as noisy. In most cases for the proposed cleansing procedure based on optimal filtering, the height of the red bar is well below the level of noise injected. This means that a significant fraction of perturbed instances are not detected. An extreme example is *Sonar*. We conjecture that this lack of sensitivity is due to the strong overlap of the distributions for the two classes. For this reason, it is difficult to single out noisy instances located in regions where such overlap is high. Using majority filtering, which is more aggressive, it is possible to detect most of the injected noisy instances. In fact, the relative performance of majority filtering improves when the levels of class-label noise are high. Still, even at high noise levels, the precision of the method is rather low: the percentage of instances that are marked as noise by majority voting is significantly larger than the level of noise injected. By contrast, even though the proposed optimal filtering procedure fails to identify some noisy instances, those that are identified by this strategy are more likely to be noise in actuality. In some datasets (*Boston*, *Breast*, *Chess*, *Parkinsons*, *Ringnorm*, *Spambase*, *Tictactoe* and *Twonorm*) the proposed procedure detects a fairly high percentage of the injected noise without removing a significant number of noiseless instances. For these datasets, random forests trained on data cleaned with the proposed procedure are more accurate than those trained based on data cleaned majority filtering, except in *Breast* and *Twonorm*, where the differences are not statistically significant (see Tables 4.2, 4.3 and 4.4).

In summary, the proposed cleansing procedure achieves high specificity at the expense of not being able to detect some noisy instances. By contrast, majority filtering detects most noisy instances, but also incorrectly discards a high percentage of valid ones. As shown in Fig. 4.2 (b), θ^*, the optimal threshold for filtering, becomes closer to 0.5 (majority filtering) as the level of class-label noise increases.

4.6 Conclusions

In this paper, we have proposed a two-stage method for the detection of class-label noise based on the robustness to noise of randomized ensembles that use resampling [Sabzevari et al., 2014, 2015]. Near-optimal values of the sampling rate can be determined using out-of-bag data. Typically, the selected sampling ratios become smaller as the level of class label noise increases. Another important observation is that the classifiers in the ensemble tend to make more errors on noisy instances. Therefore, the fraction of incorrect predictions can be used as an indicator for noise detection: if this quantity is above a threshold, θ, the instance considered is marked as noisy. Standard values for the threshold are $\theta = 0.5$ (majority) or $\theta = 1$ (consensus). In this work, we have shown that the best results are obtained at a value θ^* that is intermediate between these extremes. A simple wrapper procedure is proposed to determine θ^*. This optimal value depends on the problem under consideration and the amount of class-label noise. Values of θ^* closer to majority filtering are generally obtained for noisy problems. However, majority cleansing tends to discard instances that are correctly labeled. This has been shown to be disadvantageous, specially in problems with low levels of noise. In general, adjusting the threshold used to detect noisy instances allows us to build more accurate ensembles at all levels of class-label noise.

Once the noisy instances have been identified, they can be removed from the training data (filtering) or corrected (cleaning). Filtering is slightly superior at low and medium noise levels. Cleaning tends to be more accurate when the noise levels are high. The reason for this behavior is that filtering discards training instances. This involves some information loss, which could be detrimental if too many instances are discarded.

4.7 Acknowledgements

The authors acknowledge financial support from the *Spanish Ministry of Economy, Industry and Competitiveness*, projects TIN2013-42351-P, TIN2016-76406-P, and TIN2015-70308-REDT, and of the *Comunidad de Madrid*, project CASI-CAM-CM (S2013/ICE-2845).

Chapter 5

Vote-boosting ensembles

5.1 Abstract

Vote-boosting is a sequential ensemble learning method in which the individual classifiers are built on different weighted versions of the training data. To build a new classifier, the weight of each training instance is determined in terms of the degree of disagreement among the current ensemble predictions for that instance. For low class-label noise levels, especially when simple base learners are used, emphasis should be made on instances for which the disagreement rate is high. When more flexible classifiers are used and as the noise level increases, the emphasis on these uncertain instances should be reduced. In fact, at sufficiently high levels of class-label noise, the focus should be on instances on which the ensemble classifiers agree. The optimal type of emphasis can be automatically determined using cross-validation. An extensive empirical analysis using the beta distribution as emphasis function illustrates that vote-boosting is an effective method to generate ensembles that are both accurate and robust.

5.2 Introduction

In ensemble learning, the outputs of a collection of diverse classifiers are combined to exploit their complementarity, in the expectation that the global ensemble prediction be more accurate than the individual ones [Dietterich, 2000b]. The complementarity of the classifiers is either an indirect consequence of diversity, as in bagging [Breiman, 1996c], random forests [Breiman, 2001], and class-switching [Martínez-Muñoz and Suárez, 2005], or can be explicitly favored by design, as in negative correlation learning [Liu and Yao, 1999] and boosting [Freund and Schapire, 1997; Friedman et al., 2000; Schapire, 1990; Schapire and Freund, 2012]. In this manuscript, we present vote-boosting, an ensemble

learning method of the latter type, in which the progressive focus on a particular training instance depends on the degree of disagreement among the predictions of the ensemble classifiers. By contrast to standard boosting algorithms, the strength of this emphasis is independent of whether the instance is correctly or incorrectly classified. The optimal emphasis profile depends on the characteristics of the classification problem considered and on the complexity of the base learners. In problems with no or low levels of noise in the class labels of the training instances the appropriate focus is on instances on which the classifiers disagree (i.e. instances for which the ensemble prediction has a high degree of uncertainty). As the noise level increases, especially when more flexible base learners (e.g. unpruned CART or random trees) are used, the emphasis on uncertain instances should be reduced. In fact, in problems with sufficiently high levels of class-label noise, the optimal strategy is to assign larger weights to instances on which the ensemble classifiers agree. In practice, the determination of the optimal emphasis strategy can be automatically made through parameter selection (e.g. through cross-validation on the training data). The results of an extensive empirical are used to illustrate that vote-boosting is an effective method to build ensembles that are both accurate and robust to class-label noise.

The article is organized as follows: In Section 5.3, we provide a review of ensemble methods that are related to the current proposal. Vote-boosting is described in section 5.4. In this section, we show that the ensemble construction algorithm can be viewed as an optimization by gradient descent in the functional space of linear combinations of hypothesis. In Section 5.5, the properties and performance of vote-boosting ensembles are analyzed in an extensive empirical evaluation on synthetic and real-world classification tasks from different domains of application. Finally, the conclusions of this study are summarized in Section 5.6.

5.3 Previous Work

There are a wide variety of methods to build ensembles. In this work, we focus on homogeneous ensembles, which are composed of predictors of the same type. Each of the predictors in the ensemble is built from a training set composed of labeled instances. Once the individual predictors have been built, their outputs are combined to reach a global ensemble decision. A wide range of alternatives can be used to carry out this combination [Tulyakov et al., 2008]. Nevertheless, simple strategies, such as averaging real-valued outputs, or majority voting, if the individual classifiers yield class-labels, are generally effective [Fumera and Roli, 2005].

Randomization techniques can be used to generate collections of diverse classifiers. The objective is to build predictors that err on different examples. If the errors of these classifiers are independent, they can be averaged out by the combination process. An example of these types of ensembles is bagging [Breiman, 1996c]. The individual classifiers in a bagging ensemble are built by applying a fixed learning algorithm to independent bootstrap samples drawn from the original training data. In class-switching ensembles, each member is built using a perturbed version of the original training set, in which the class labels of a fraction of instances are modified at random [Breiman, 2000; Martínez-Muñoz and Suárez, 2005]. Alternatively, diverse classifiers can be built by including some randomized steps in the learning algorithm itself. For instance, in one of the earliest works on ensembles [Hansen and Salamon, 1990], one takes advantage of the presence of multiple local minima in the optimization process used to determine the synaptic weights of a multilayer perceptron. Starting from different initial seeds, it is possible to build neural networks with the same architecture, but different weights. Each of these different networks yields a different prediction. The final ensemble prediction can be obtained by pooling the individual decisions of the neural networks that result from the different weight initializations. Random forests [Breiman, 2001], which are one of the most effective ensemble methods [Fernández-Delgado et al., 2014b], are built using a combination of data randomization and randomization in the learning algorithm: The ensemble classifiers are random trees trained on bootstrap replicates of the original training dataset. The individual ensemble trees are generated using the random subspace method [Ho, 1998]. Other effective classifiers of this type are rotation forests [Rodríguez et al., 2006], ensembles of extremely randomized trees [Geurts et al., 2006], and other variants of random forest [Younsi and Bagnall, 2016; Yu et al., 2016].

An alternative to simply generating diversity is to explicitly aim to increase the complementarity of the ensemble classifiers. An example of such strategy is Negative Correlation Learning [Liu and Yao, 1999]. In this method, complementarity is favored by simultaneously training all the classifiers in the ensemble: The parameters of the individual classifiers and the weights of the combination of their outputs are determined globally by minimizing a cost function that penalizes coincident predictions. One can also build ensembles of base learners that are trained to focus on different regions in feature space [Armano and Tamponi, 2018]. Boosting is another ensemble method in which complementarity among the classifiers is explicitly favored. Boosting originally refers to the problem of building a strong learner out of a collection of weak learners; i.e. learners whose predictive accuracy is only slightly better than random guessing [Friedman et al., 2000; Schapire, 1990; Schapire and Freund, 2012]. AdaBoost is one of the most widely used boosting algorithms [Freund and Schapire, 1997]. In its original formulation, AdaBoost considered only binary classification tasks. Nonetheless, there

are numerous extensions to deal with multiclass problems (see e.g. the references in [Fernández-Baldera and Baumela, 2014]). In AdaBoost an ensemble is grown by incorporating classifiers that progressively focus on instances that are misclassified by the previous classifiers in the sequence. The individual classifiers are built by applying a learning algorithm that can handle individual instance weights. Alternatively, weighted resampling in the training set can be used. The first classifier is obtained by assuming equal weights for all instances. The subsequent classifiers are built using different emphasis on each of the training instances. Specifically, to build the t-th classifier in the sequence, the weights of instances that are misclassified by the most recent classifier in the ensemble are increased. Correspondingly, the weights of the correctly classified instances are reduced. The final prediction of the ensemble is determined by weighted majority voting. The weight of an individual classifier in the final ensemble prediction depends on the weighted accuracy of this classifier on the training set. The margin of an instance is defined as the sum of weighted votes for the correct class minus the sum of weighted votes for the most voted incorrect class. Therefore, misclassified instances have negative margins. In AdaBoost, the evolution of the weight of a particular instance is a monotonically decreasing function of its margin [Freund et al., 1999].

AdaBoost is one of the most effective ensemble methods [Dietterich, 2000a; Fernández-Delgado et al., 2014b; Opitz and Maclin, 1999]. However, it is not robust to class-label noise [Bauer and Kohavi, 1999; Khoshgoftaar et al., 2011; Quinlan, 1996]. Specifically, AdaBoost gives unduly high weights to noisy instances, whose class labels are incorrect. There are numerous studies that address this excessive sensitivity of AdaBoost to class-label noise [Abe et al., 2006; Cao et al., 2012; Cheamanunkul et al., 2014; Domingo and Watanabe, 2000; Freund, 2001, 2009; Friedman et al., 2000; Gómez-Verdejo et al., 2008; Guo et al., 2002; Jiang, 2001; Loureiro et al., 2004; Rätsch et al., 1998; Shivaswamy and Jebara, 2011; Sun et al., 2004, 2006]. A possible strategy is to identify and either remove noisy instances in the training data, or correct their class-labels [Abe et al., 2006; Gao and Gao, 2010; Loureiro et al., 2004]. Modified weight update rules can be used for instances that are identified as noisy [Cao et al., 2012; Sun et al., 2016]. Another alternative is to apply explicit or implicit regularization techniques to avoid assigning excessive weight to a reduced group of instances [Domingo and Watanabe, 2000; Gómez-Verdejo et al., 2008; Gómez-Verdejo et al., 2010; Jiang, 2001; Mayhua-Lopez et al., 2012; Shivaswamy and Jebara, 2011]. For instance, the logistic loss function employed in LogitBoost [Friedman et al., 2000] gives less emphasis to instances with large negative margins than the exponential loss function used in AdaBoost. In consequence, LogitBoost is generally more robust to class-label noise [McDonald et al., 2003]. In other studies, penalty terms are used in the cost function to avoid focusing on outliers or on instances that are difficult to classify [Guo et al., 2002; Rätsch et al., 1998; Sun

et al., 2004, 2006]. It is possible to also use hybrid weighting methods that modulate the emphasis on instances according to their distance to the decision boundary [Ahachad et al., 2015, 2017; Gómez-Verdejo et al., 2006, 2008]. Most boosting algorithms use convex loss functions. This has the advantage that the resulting optimization problem can be solved efficiently using, for instance, gradient descent. However, as shown in [Long and Servedio, 2010], the generalization capacity of boosting variants that use convex loss functions can be severely affected by class-label noise. Alternative non-convex loss functions are used in BrownBoost and other robust boosting variants [Cheamanunkul et al., 2014; Freund, 2001, 2009; Masnadi-shirazi and Vasconcelos, 2009; Miao et al., 2016]. In these methods, the evolution of the weights is *not* a monotonic function of the margin: instances with small negative margins (i.e. misclassified instances that are close to the decision boundary) are assigned higher weights, as in AdaBoost. However, instances whose margin is negative and large receive lower weights. The rationale for using this type of emphasis is that instances in regions with a large class overlap tend to have small margins. Focusing on these instances is beneficial because the classification boundary can be modeled in more detail. By contrast, large negative margins correspond to misclassified instances that are far from the classification boundary. A robust boosting algorithm should therefore avoid emphasizing these instances, which are likely to be noisy.

In the next section we introduce vote-boosting, a novel boosting algorithm in which the weights of the instances are determined in terms of the degree of agreement or disagreement among the predictions of the ensemble members, irrespective of their actual class labels. As illustrated by the results of empirical evaluation presented in section 5.5, the optimal type of emphasis (that is, whether the focus should be placed on instances on which the classifiers disagree, or on instances on which they agree) can be determined from the training data alone using, for instance, cross-validation. Since the instance weights do not depend on whether the predictions by the ensemble classifiers are correct, it is possible to avoid unduly emphasizing incorrectly classified instances that are outliers, which is one of the weaknesses of standard boosting algorithms, such as AdaBoost. In this manner, one can build accurate ensembles that are robust to class-label nosie.

5.4 Vote-boosting

Consider the problem of automatic induction of a classification system from labeled data. The original training set is composed of N_{train} attribute class-label pairs $\mathcal{D}_{train} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N_{train}}$, where $\mathbf{x} \in \mathcal{X}$. In this article, we focus on binary classification tasks, in which $y \in \{-1, 1\}$. Problems with multiple classes can be addressed with a number

of strategies, such as the ones used in combination with AdaBoost for this purpose [Schapire and Freund, 2012].

Let $\{f_\tau(\cdot)\}_{\tau=1}^{t}$ be a partially-grown ensemble of size t. The τ-th classifier in the ensemble is a function $f_\tau : \mathcal{X} \to \{-1, 1\}$ that maps a vector of attributes $\mathbf{x} \in \mathcal{X}$ to a class label $f_\tau(\mathbf{x}) \in \{-1, 1\}$. This function is obtained by applying a base learning algorithm to a training set, taking into account the individual instance weights $\left\{w_i^{[\tau]}\right\}_{i=1}^{N_{train}}$.

To obtain the prediction of the ensemble, the predictions of the individual classifiers are aggregated by weighted averaging

$$F_t(\mathbf{x}) = \sum_{\tau=1}^{t} \alpha_\tau^{[t]} f_\tau(\mathbf{x}), \quad F_t \in [-1, 1]\,, \tag{5.1}$$

where $\alpha_\tau^{[t]} \geq 0$ is the weight of the prediction of the τ classifier. These weights are normalized $\sum_{\tau=1}^{t} \alpha_\tau^{[t]} = 1$. In (unweighted) majority voting one assumes that all the predictions have the same weight , $\alpha_\tau^{[t]} = 1/t$. This simple voting scheme provides good overall results. For this reason, it will be used in our implementation. Based on this aggregated output, the final prediction of the ensemble of size t is

$$H_t(\mathbf{x}) = sign\left(F_t(\mathbf{x})\right). \tag{5.2}$$

Assuming that t is odd, $H_t(\mathbf{x}) \in \{-1, 1\}$. At this stage of the ensemble construction process, instance $\mathbf{x}$ can be characterized by $t_+(\mathbf{x})$ and $t_-(\mathbf{x}) = t - t_+(\mathbf{x})$, the counts of positive and negative votes, respectively. The fractions of votes in each class are

$$\pi_\pm^{[t]}(\mathbf{x}) = \frac{t_\pm(\mathbf{x})}{t}; \quad \pi_+^{[t]}(\mathbf{x}) + \pi_-^{[t]}(\mathbf{x}) = 1. \tag{5.3}$$

These values can be used to quantify the level of certainty of the ensemble prediction. Values $\pi_+^{[t]}(\mathbf{x})$ close to 0 or 1 correspond to instances for which the predictions of most ensemble classifiers coincide. Instances whose classification by the ensemble is uncertain are characterized by $\pi_+^{[t]}(\mathbf{x})$ close to 1/2. In contrast to standard boosting algorithms, in vote-boosting, the instance weights depend on the degree of agreement or disagreement among the predictions of the individual classifiers, not on whether these predictions are erroneous. The pseudo-code of the proposed vote-boosting algorithm is presented in Algorithm 5.

The final ensemble is composed of T classifiers, each of which is built by applying the base learning algorithm $\mathcal{L}$ on the training data with different sets of instance weights. The weights of the instances can be taken into account using weighted resampling with

Algorithm 5: Vote-boosting algorithm with resampling

Input:

$\mathcal{D}_{train} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N_{train}}$, $\mathbf{x}_i \in \mathcal{X}$, $y_i \in \{-1, 1\}$

T % Ensemble size

$\mathcal{L}$ % Base learning algorithm

$g(p)$ % Non-negative emphasis function ($0 \leq p \leq 1$)

$F_0(\cdot) \leftarrow 0$

$t_+(\mathbf{x}_i) \leftarrow 0 \ \forall i = 1, ..., N_{train}$

$w_i^{[1]} \leftarrow \frac{1}{N_{train}} \ \forall i = 1, ..., N_{train}$

for $t \leftarrow 1$ ***to*** T **do**

  $f_t(\cdot) \leftarrow \mathcal{L}\left(\mathcal{D}_{train}, \mathbf{w}^{[t]}\right)$

  $t_+(\mathbf{x}_i) \leftarrow t_+(\mathbf{x}_i) + \mathbb{I}(f_t(\mathbf{x}_i) > 0) \ \forall i = 1, ..., N_{train}$

  $\pi_+^{[t]}(\mathbf{x}_i) = \frac{t_+(\mathbf{x}_i)+1}{t+2} \ \forall i = 1, ..., N_{train}$

  $w_i^{[t+1]} \leftarrow g\left(\pi_+^{[t]}(\mathbf{x}_i)\right) \ \forall i = 1, ..., N_{train}$

  Normalize $\mathbf{w}^{[t]}$

  $F_t(\cdot) \leftarrow F_{t-1}(\cdot) + f_t(\cdot)$

Output: $H_T(\cdot) = sign(F_T(\cdot))$

replacement

$$\begin{aligned} \mathcal{D}_{train}^{[t]} &= \text{Sample}\left(\mathcal{D}_{train}, \mathbf{w}^{[t]}\right) \\ f_t(\cdot) &\leftarrow \mathcal{L}\left(\mathcal{D}_{train}^{[t]}\right). \end{aligned} \tag{5.4}$$

For the induction of the first ensemble classifier all instances are assigned the same weight. These weights are updated at each iteration according to the tally of votes: Assuming that instance $\mathbf{x}_i$ has received $t_+(\mathbf{x}_i)$ votes for the positive class at the t-th iteration, we use the Laplace estimator of the probability that a classifier in the ensemble outputs a particular class prediction

$$\pi_\pm^{[t]}(\mathbf{x}_i) = \frac{t_\pm(\mathbf{x}_i) + 1}{t + 2}. \tag{5.5}$$

Finally the weights are updated

$$w_i^{[t+1]} = \frac{1}{\mathcal{Z}_{t+1}} g\left(\pi_+^{[t]}(\mathbf{x}_i)\right), \quad \text{for } \ i = 1, ..., N_{train}, \tag{5.6}$$

where $g : [0, 1] \to \mathbb{R}^+$ is an emphasis function, which is non-negative, and

$$\mathcal{Z}_{t+1} = \sum_{i=1}^{N_{train}} g\left(\pi_+^{[t]}(\mathbf{x}_i)\right) \tag{5.7}$$

is a normalization constant. These weights are then used to build the following classifier in the ensemble.

A natural choice for the emphasis function is the probability density of the beta distribution with shape parameters a, b

$$g(p) \equiv \beta\,(p; a, b) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} p^{a-1}(1-p)^{b-1}, \quad 0 \leq p \leq 1. \tag{5.8}$$

For this particular emphasis function, the weights at the $(t+1)$-th iteration are updated according to

$$w_i^{[t+1]} = \frac{1}{Z_{t+1}} \beta \left(\pi_+^{[t]}(\mathbf{x}_i); a, b \right) \tag{5.9}$$

If the class distributions are not strongly imbalanced, the choice $a = b$, in which the two classes are handled in a symmetrical manner, is generally appropriate. In problems with a large class imbalance, an asymmetric choice of the emphasis function may be preferable. In Figure 5.1 the density profiles of the symmetric beta distribution for different values of $a = b \in \{0.25, 0.75, 1, 1.5, 2.5, 5, 10, 20, 40\}$ are shown. If $a = b = 1.0$, the distribution is uniform. Therefore, all instances are given the same importance (plot in the first row, third column of Figure 5.1). In this case, the proposed algorithm is equivalent to bagging [Breiman, 1996c]. For $a = b > 1.0$ the distribution becomes unimodal, with a maximum at 0.5. In this range, the higher the values of $a = b$, the more concentrated becomes the probability around the mode. In consequence, vote-boosting emphasizes uncertain instances and reduces the importance of those instances on which most classifiers agree. For simple classifiers, the emphasis on uncertain instances that one obtains is similar to the error-based emphasis of AdaBoost. The reason is that uncertain instances are generally more difficult to classify and, in consequence, are more likely to be incorrectly classified. In the regime $a = b < 1.0$, vote-boosting progressively focuses on instances on which classifiers agree. As will be illustrated in the section on experiments, this strategy is effective in complex or noisy problems, especially when the ensemble is composed of flexible classifiers, because of its regularizing effects.

It is common, especially in the first iterations of the algorithm, when the ensemble is still small, that all the classifiers predict the same class label for some instances. In such cases, the fraction of positive votes is either 0 or 1. Except for $a = b = 1$, the value of the beta distribution at these points is either zero or infinity. In the case of zero density values, those instances would be assigned zero weight in the next iteration of the algorithm. Thus, they would be effectively removed from the sample. In the other extreme, some instances would have infinite weights. To avoid these evaluations of the beta distribution at the boundaries of its support, the Laplace correction has been used in the estimation of class prediction probabilities (5.5).

Finally, we note that vote-boosting can be used with base learners that achieve zero or low error rates in the training data. In such cases, the weights that AdaBoost assigns

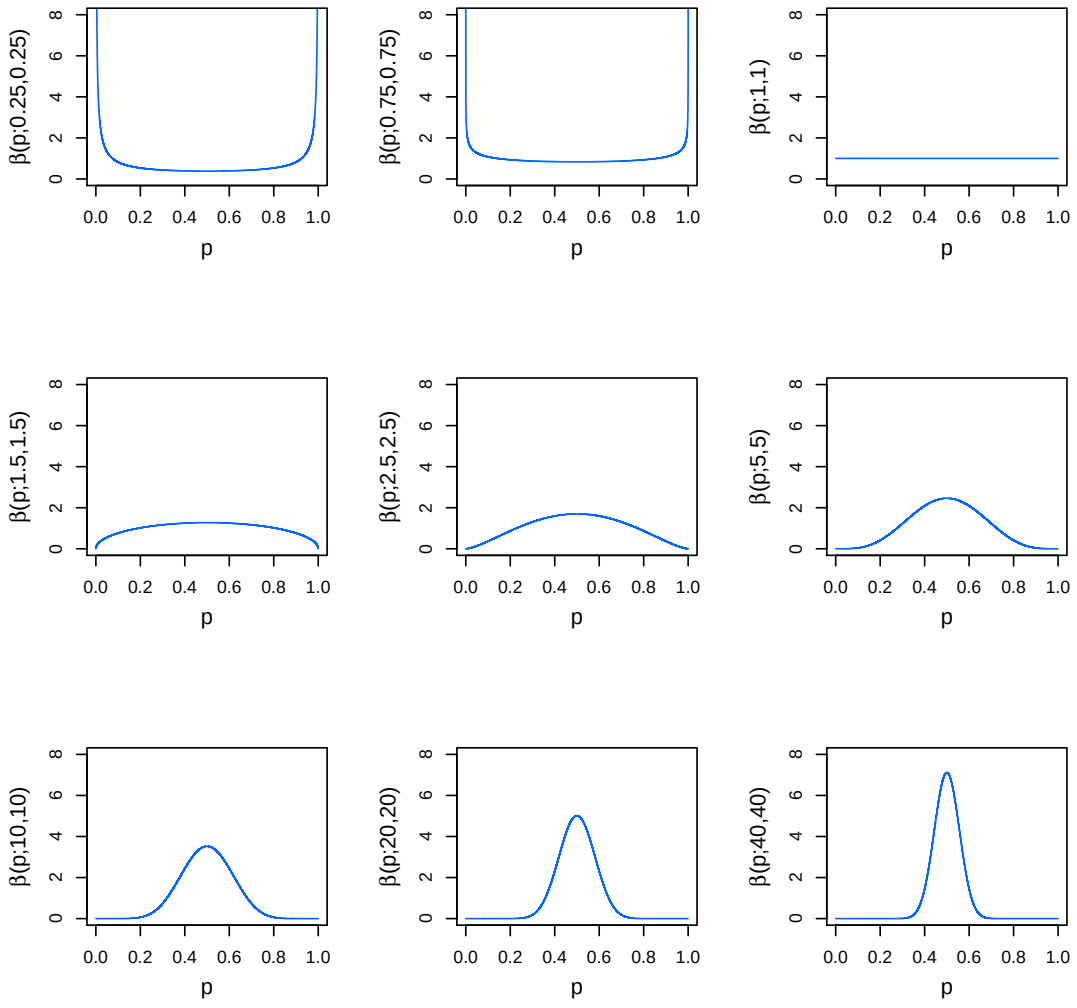


Fig. 5.1. Symmetric beta distribution with $a = b = [0.25, 0.75, 1, 1.5, 2.5, 5, 10, 20, 40]$

to the training instances are ill-defined. By contrast, even if the training error of a single ensemble classifier is zero or close to zero, there can be disagreement among the individual predictions. Therefore, provided that the Laplace correction is used in the estimation of class prediction probabilities, the weights given by Eq. 5.9, are always well defined. This feature allows us to build vote-boosting ensembles of unpruned CART or random trees, which, as illustrated by the results of Section 5.5 are both accurate and robust to class-label noise.

5.4.1 An interpretation of vote-boosting as functional gradient descent

Similarly to other boosting methods, vote-boosting can be viewed as a gradient descent algorithm in the hypothesis space of linear combinations of predictors [Mason et al.,

2000]. Consider an ensemble of t predictors $\{f_\tau\}_{\tau=1}^{t}$. The global ensemble prediction on instance $\mathbf{x}$ is of the form

$$H_t(\mathbf{x}) = sign[F_t(\mathbf{x})], \tag{5.10}$$

where

$$F_t(\mathbf{x}) = \frac{1}{t}\sum_{\tau=1}^{t} f_\tau(\mathbf{x}), \quad F_t(\mathbf{x}) \in [-1,1]. \tag{5.11}$$

The fraction of votes for the positive class can be expressed in terms of this quantity

$$\pi_+^{[t]}(\mathbf{x}) = \frac{1+F_t(\mathbf{x})}{2}. \tag{5.12}$$

Consider the predictor $F : \mathbf{x} \in \mathcal{X} \rightarrow F(x) \in [-1,1]$. Let the cost functional for this predictor be

$$C[F] = \frac{1}{N_{train}}\sum_{i=1}^{N_{train}} y_i c(F(\mathbf{x}_i)), \tag{5.13}$$

where $c(z)$ is a monotonically non-increasing function of $z \in [-1,1]$, such that $c(0) = 0$. These properties ensure that when the prediction of F for the i-th example is class -1 (i.e. $F(\mathbf{x}_i) < 0$), then $c(F(\mathbf{x}_i)) > 0$. Similarly, when the prediction is class $+1$ (i.e. $F(\mathbf{x}_i) > 0$), then $c(F(\mathbf{x}_i)) < 0$. Therefore, when the prediction $F(\mathbf{x}_i)$ is incorrect (i.e. $y_i F(\mathbf{x}_i) < 0$), the contribution to the cost functional $y_i c\left(F(\mathbf{x}_i)\right)$ is positive. When the prediction is correct, the corresponding contribution is negative. From these properties ones concludes that $C[F]$ achieves its global minimum when the training error is zero. Furthermore, this quantity increases with each incorrect prediction. In consequence, the minimizer of $C[F]$ also minimizes the error of predictor F in the training set.

If $|F(\mathbf{x}_i)|$ is a measure of how certain the prediction of F for instance $\mathbf{x}_i$ is, the value $|c(F(\mathbf{x}_i))|$ provides also a measure of such certainty. The magnitude of contribution $y_i c\left(F_t(\mathbf{x}_i)\right)$ to the cost functional increases with the margin of the prediction. It is largest when $|F_t(\mathbf{x}_i)| = 1$; that is, when all ensemble classifiers agree.

In vote-boosting, the first classifier in the ensemble is built by assuming equal weights for all the instances in the training set. Then, the ensemble is grown in a sequential manner by incorporating to $\{f_\tau\}_{\tau=1}^{t}$, the ensemble of size t, the classifier that minimizes the value of cost functional for the enlarged ensemble $\{f_\tau\}_{\tau=1}^{t} \cup \{f_{t+1}\}$

$$f_{t+1} = \underset{f\in\mathcal{F}}{\arg\min}\, C\left[F_{t+1}^{[f]}\right], \tag{5.14}$$

where

$$F_{t+1}^{[f]}(\mathbf{x}) = \frac{1}{t+1}\sum_{\tau=1}^{t} f_\tau(\mathbf{x}) + \frac{1}{t+1} f(\mathbf{x}) = F_t(\mathbf{x}) + \frac{1}{t+1}\left(f(\mathbf{x}) - F_t(\mathbf{x})\right). \tag{5.15}$$

Assuming the change in the value of the cost functional when the ensemble incorporates the new classifier f is small

$$\begin{aligned}
\delta C[F_t] &\equiv C\left[F_{t+1}^{[f]}(\mathbf{x})\right] - C\left[F_t(\mathbf{x})\right] \\
&= \frac{1}{N_{train}} \sum_{i=1}^{N_{train}} y_i \left[c\left(F_t(\mathbf{x}_i) + \frac{1}{t+1}\left(f(\mathbf{x}_i) - F_t(\mathbf{x}_i)\right)\right) - c\left(F_t(\mathbf{x}_i)\right)\right] \\
&\approx \frac{1}{(t+1)N_{train}} \left[\sum_{i=1}^{N_{train}} y_i c'(F_t(\mathbf{x}_i)) f(\mathbf{x}_i) - \sum_{i=1}^{N_{train}} y_i c'(F_t(\mathbf{x}_i)) F_t(\mathbf{x}_i)\right],
\end{aligned} \tag{5.16}$$

to lowest order in the Taylor expansion. The second term in the last expression does not depend on f. Therefore, to lowest order, minimizing the cost functional is equivalent to minimizing

$$\begin{aligned}
&\sum_{i=1}^{N_{train}} y_i c'(F_t(\mathbf{x}_i)) f(\mathbf{x}_i) \\
&\quad = \sum_{i:y_i = f(x_i)} c'(F_t(\mathbf{x}_i)) - \sum_{i:y_i \neq f(x_i)} c'(F_t(\mathbf{x}_i)) \\
&\quad = \sum_{i=1}^{N_{train}} c'(F_t(\mathbf{x}_i)) - 2 \sum_{i:y_i \neq f(x_i)} c'(F_t(\mathbf{x}_i)) \\
&\quad = 2 \sum_{j=1}^{N_{train}} c'(F_t(\mathbf{x}_j)) \left(\frac{1}{2} - \sum_{i:y_i \neq f(x_i)} w_i^{[t+1]}\right),
\end{aligned}$$

where

$$w_i^{[t+1]} = \frac{c'\left(F_t(\mathbf{x}_i)\right)}{\sum_{j=1}^{N_{train}} c'\left(F_t(\mathbf{x}_j)\right)}. \tag{5.17}$$

Since $c(z)$ is a monotonic non-increasing function, then $-c'(z)$ is non-negative, and the values $\left\{w_i^{[t+1]}\right\}_{i=1}^{N_{train}}$ defined in Eq. (5.17) can be thought of as a set of instance weights. The denominator in (5.17) ensures that these weights are normalized

$$\sum_{i=1}^{N_{train}} w_i^{[t+1]} = 1. \tag{5.18}$$

Using this expression for the instance weights, Eq. (5.16) becomes

$$\delta C[F_t] \propto \sum_{i:y_i \neq f(x_i)} w_i^{[t+1]} - \sum_{i:y_i \neq F_t(x_i)} w_i^{[t+1]}. \tag{5.19}$$

Note that $\delta C[F_t] < 0$ only if the weighted training error of the newly built classifier is lower than the corresponding error for the ensemble.

Under these conditions, the $(t+1)$-th predictor in the ensemble is the minimizer of the *weighted* training error

$$f_{t+1} = \arg\min_{f \in \mathcal{F}} \sum_{i: y_i \neq f(x_i)} w_i^{[t+1]}, \tag{5.20}$$

where $\mathcal{F}$ is the functional space of the base learners. In contrast to standard boosting algorithms, the weights given by (5.17) depend only on the ensemble predictions, irrespective of whether these predictions are correct.

Assuming that the function $c(z)$ in (5.13) is bounded, it is convenient to express it in terms of a cumulative distribution function $G(\pi)$ defined in the unit interval $\pi \in [0,1]$

$$c\left(F(\mathbf{x})\right) \;=\; 2K\left[G(1/2) - G\left(\frac{1+F(\mathbf{x})}{2}\right)\right], \tag{5.21}$$

where K is a positive constant. Without loss of generality, this constant is set to one ($K = 1$). Because of the monotonicity of $G(p)$, when the prediction $F(\mathbf{x}_i)$ is incorrect (i.e. $y_i F(\mathbf{x}_i) < 0$), the contribution to the cost functional $y_i c\left(F(\mathbf{x}_i)\right)$ is positive. Assuming this form for $c\left(F(\mathbf{x})\right)$, its derivative is

$$c'\left(F(\mathbf{x})\right) \;=\; -g\left(\frac{1+F(\mathbf{x})}{2}\right), \tag{5.22}$$

where $g(p) = G'(p)$ is the corresponding probability density, which is non-negative. With these assumptions, the weights of the training instances are

$$w_i^{[t+1]} = \frac{g\left(\pi_+^{[t]}(\mathbf{x}_i)\right)}{\sum_{j=1}^{N_{train}} g\left(\pi_+^{[t]}(\mathbf{x}_j)\right)}, \quad i = 1, 2, \ldots, N_{train}, \tag{5.23}$$

where the density $g(p)$ plays the role of an emphasis function. Note that this density need not be symmetric around $\pi_+^{[t]}(\mathbf{x}_i) = \frac{1}{2}$. In fact, asymmetries in the emphasis could be useful in classification problems with unbalanced classes. In contrast to most boosting algorithms, including AdaBoost, the weights given by Eq. (5.23) do not depend on the actual class label of the instance. Therefore, it does not seem possible to derive error bounds similar to those enunciated in Theorem 6 (e.g. Eq. (21)) of [Freund and Schapire, 1997].

5.5 Empirical evaluation

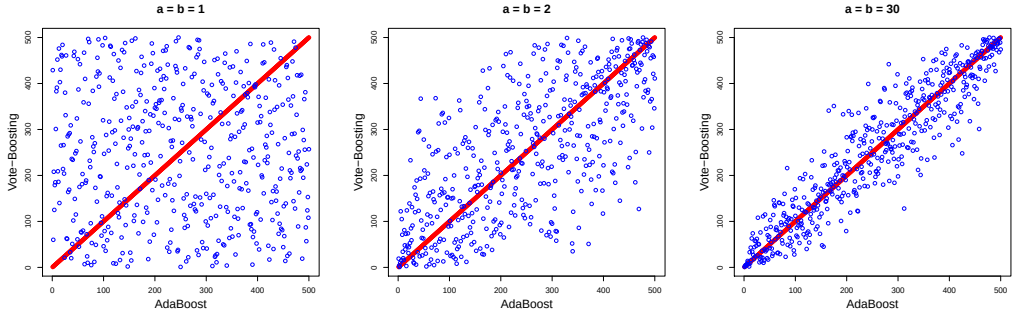
In this section we present the results of an empirical analysis of vote-boosting. Different sets of experiments have been performed to analyze the properties ensembles built with this method and evaluate their accuracy in a wide range of classification tasks from

different areas of application. In these experiments, the symmetric beta distribution has been used as the emphasis function. A first set of experiments is carried out to investigate the relationship of vote-boosting with bagging and AdaBoost. The results of these experiments illustrate that, when simple (e.g. decision stumps) or regularized learners (e.g. puned CART trees) are used as base learners, vote-boosting performs an interpolation between bagging ($a = b = 1.0$) and AdaBoost (high values of $a = b$). In a second set of experiments, we investigate the behavior of vote-boosting composed of different classifiers. In particular, we compare the accuracies of vote-boosting ensembles composed of decision stumps, pruned CART trees, unpruned CART trees, and (unpruned) random trees. The best overall results in terms of predictive accuracy are obtained with random trees. However, the differences with vote-booting ensembles composed of pruned or unpruned CART trees are not statistically significant. Finally, the accuracy of vote-boosting ensembles composed of random trees is compared with bagging, AdaBoost and random forest. From the results of this benchmarking exercise, we conclude that, in the problems investigated, vote-boosting ensembles composed of random trees achieve state-of-the-art classification accuracy rates. These rates are comparable or superior to random forest and AdaBoost. A final batch of experiments is carried out to analyze the differences among the optimal emphasis profiles for different classification problems using random trees as base learners. This analysis illustrates that in problems with low levels of noise in the class labels, new classifiers should focus on instances whose classification by the current ensemble is uncertain By contrast, in problems with contaminated labels, the optimal emphasis is to reduce the weights of such uncertain instances, which are likely to be noisy.

5.5.1 Vote-boosting as an interpolation between bagging and AdaBoost

The objective of the experiments presented in this subsection is to analyze how the behavior of vote-boosting ensembles composed of simple or regularized learners, such as decision stumps, or pruned CART trees, changes when different levels of emphasis on the uncertain training instances are considered. As discussed earlier, when uniform emphasis is made, vote-boosting is equivalent to bagging. In most of the problems analyzed, when such simple base learners are used, stronger emphasis on uncertain instances (i.e. instances for which the degree of disagreement among the ensemble predictions is largest) results in a behavior that is similar to AdaBoost. In such cases, vote-boosting provides an interpolation between bagging and AdaBoost, depending on the strength of the emphasis on uncertain instances.

To investigate this relationship between vote-boosting and AdaBoost, we first present the results of an experiment in the binary classification problem *Twonorm* using decision

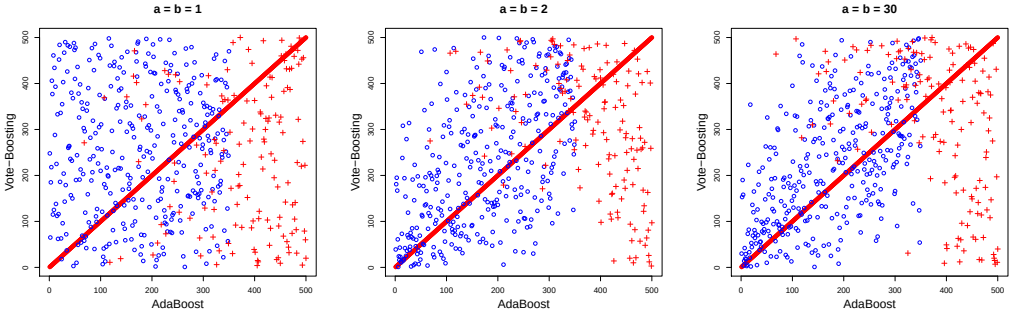


(a) **(b)** **(c)**

Fig. 5.2. Weight ranks of the training instances for vote-boosting and AdaBoost of decision stumps in *Twonorm* (a) $a = b = 1.0$, (b) $a = b = 2.0$, (d) $a = b = 30.0$

stumps as base learners. In *Twonorm*, instances are drawn from two unit-variance Gaussians in 20 dimensions whose means are $(a, a, \ldots, a)$ for class 1 and $(-a, -a, \ldots, -a)$ for class 2, with $a = 2/\sqrt{20}$ [Breiman, 1996d]. This is not a trivial task for decision stumps because, as individual classifiers, they can model only class boundaries that are parallel to the axes. In the experiments performed, the training set is composed of 500 independently generated instances. Different vote-boosting ensembles composed of 100 stumps were built using the symmetric beta distribution for emphasis, with $a = b \in \{1.0, 2.0, 30.0\}$. An AdaBoost ensemble composed of 100 stumps was also built using the same training data. The final weights given to the training instances in the different ensembles were recorded and subsequently ranked. Ties were resolved by randomizing the corresponding ranks. In Figure 5.2, a scatter plot of these ranks is shown. The position along the horizontal axis corresponds to the rank assigned by AdaBoost. Correspondingly, the location along the vertical axis corresponds to the ranks assigned by vote-boosting with $a = b = 1.0$ in (a), $a = b = 2.0$ in (b), and $a = b = 30.0$ in (c). When $a = b = 1.0$ is used, all instances are assigned the same weight and no correlations between the weight ranks given by vote-boosting and AdaBoost is observed. Therefore, the points corresponding to the training instances appear uniformly distributed in Figure 5.2 (a). As the values of $a = b$ increase, points tend to cluster around the diagonal. This is a consequence of the fact that the ranks of the weights given by both types of ensembles become more similar. Comparing Figs. 5.2 (a), (b) and (c), it is apparent that the correlations between the weight ranks become stronger as $a = b$ increases. In particular for $a = b = 30.0$ vote-boosting and AdaBoost give similar emphasis, even though the former does not make use of class labels to decide whether an instance should be given more weight, whereas the latter does. The reason for this coincident emphasis is that, in this simple problem, the ensemble classifiers are more likely to disagree precisely in the instances that are incorrectly classified.

If class-label noise is injected in the problem, the weighting schemes of AdaBoost and

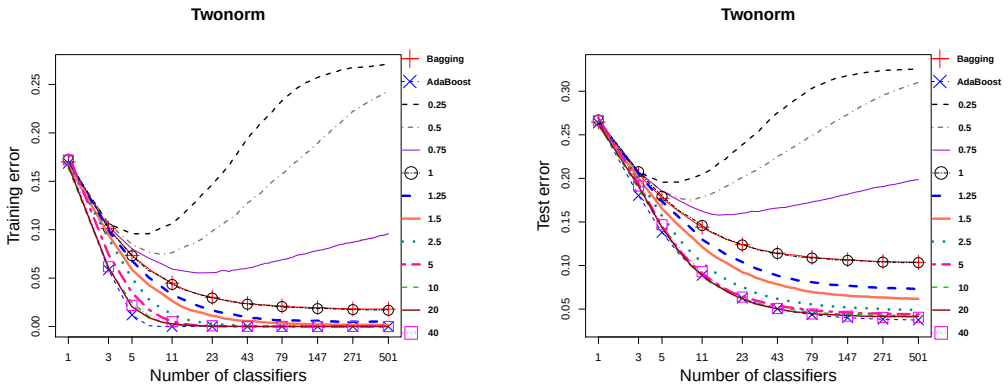


(a) **(b)** **(c)**

Fig. 5.3. Weight ranks of the training instances for vote-boosting and AdaBoost of decision stumps in *Twonorm* with 30% class-label noise, (a) $a = b = 1.0$, (b) $a = b = 2.0$, (c) $a = b = 30.0$

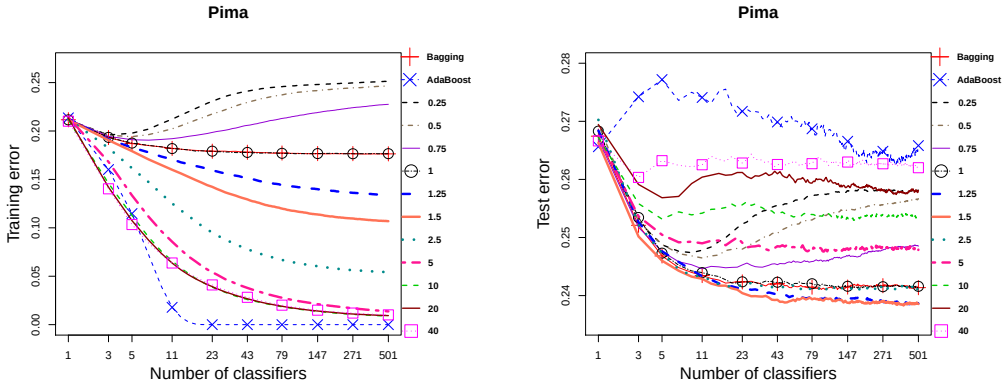
vote-boosting become different: Vote-boosting maintains the focus on instances in the boundary region, in which classes overlap and the disagreement rates among the ensemble predictions are highest. By contrast, AdaBoost tends to give more weight to those instances whose class label has been modified. Focusing on these noisy instances is misleading and eventually impairs the generalization capacity of AdaBoost. To illustrate this observation, the experiment was repeated injecting class-label noise in the training data. Specifically, 30% of the training examples were selected at random and their class labels flipped, as in the noisy completely at random model described in [Frénay and Verleysen, 2014]. The results of these experiments are shown in Figure 5.3. Instances whose class label has been switched are marked with a red cross in these plots. For $a = b = 1.0$, no correlation is observed between the ranks of the weights given by vote-boosting and AdaBoost. However, instances whose class label has been switched, which are distributed uniformly in the vertical direction, tend to appear on the right-hand side of the plots. This means that they receive special emphasis in AdaBoost but not in vote-boosting. Increasing the value of $a = b$ pushes the unperturbed instances towards the diagonal but not the perturbed ones. However, the correlation between the ranks is less marked than in the noiseless case, because of the interference of the noisy instances.

A second batch of experiments was carried out to analyze the behavior of vote-boosting composed of pruned CART trees as a function of the strength of the emphasis that is applied to uncertain instances. Using the symmetric beta distribution as emphasis function, we analyze how the learning curves, which trace the dependence of the error as a function of the size of the ensemble, depend on the value of the shape parameter. The values explored are $a{=}b \in \{0.25, 0.5, 0.75, 1.0, 1.25, 1.5, 2.5, 5, 10, 20, 40\}$. The experiments were made on the classification tasks *Twonorm* and *Pima*. These tasks have been chosen because of the different prediction accuracies of bagging and AdaBoost on



(a) **(b)**

Fig. 5.4. Error rate as a function of number of classifiers in *Twonorm* for bagging, AdaBoost, and vote-boosting ensembles of pruned CART trees: (a) training error (b) test error.



(a) **(b)**

Fig. 5.5. Error rate as a function of the number of classifiers in *Pima* for bagging, AdaBoost, and vote-boosting ensembles of pruned CART trees: (a) training error (b) test error.

those datasets. In *Twonorm*, AdaBoost significantly outperforms bagging. By contrast in *Pima*, which is a very noisy task, bagging is more accurate than AdaBoost.

Figure 5.4 and 5.5 display the learning curves for bagging, AdaBoost, and vote-boosting using different values of the shape parameter in the classification tasks *Twonorm* and *Pima*, respectively. The plots on the left-hand side show the results for the training error. The plots on the right-hand side correspond to the test error curves. When $a = b = 1.0$, all instances are given equal weights. In this case, if weighted resampling is used, vote-boosting is equivalent to bagging. This is apparent from Figure 5.4 and 5.5:

the error curves of both methods are very close to each other. When values $a = b < 1.0$ are used, emphasis is made on instances on which most predictors agree. If regularized classifiers, such as pruned CART trees, are used as base learners, this type of emphasis is in general not effective. Nonetheless, as will be illustrated by the results presented in Section 5.5.3, focusing on these types of instances can lead to improvements in the generalization capacity when the data are very noisy and the ensemble is composed of flexible classifiers that overfit (e.g. unpruned CART or random trees).

Values of $a = b > 1.0$ correspond to emphasizing instances in which the ensemble prediction is uncertain. For *Twonorm*, the learning curves of vote-boosting using $a = b = 40.0$ and AdaBoost are quite similar. In this problem, the classification errors are mainly due to the overlap between the distributions of the two classes. Therefore, the incorrectly classified instances are close to the decision boundary, where the ensemble predictions are also more uncertain. Using a beta distribution sharply peaked around $\pi = 0.5$ (see Figure 5.1) gives more weight to these uncertain instances. In consequence, the resulting emphasis is similar to AdaBoost's. In *Pima*, which is a noisy problem, the learning curves of vote-boosting with large $a = b$ and AdaBoost are different. Still, the closest performance to AdaBoost is vote-boosting with large $a = b$. Finally, we observe that the optimal value for the shape parameter of the beta distribution in vote-boosting is problem dependent: Values of $a = b \approx 1.25 - 1.5$ perform well in *Pima*. The best performance in *Twonorm* requires using a large values of $a = b \approx 40$. For each problem, the optimal value can be determined using cross-validation.

5.5.2 Vote-boosting with different base learners

In this section, we present the results of a comparison of vote-boosting ensembles composed of different base learners. These are, in order of increasing complexity, decision stumps, pruned and unpruned CART trees, and unpruned random trees. Ensembles composed of 501 classifiers are built. This fairly large ensemble size is needed in some problems to achieve convergence to the asymptotic error level [Hernández-Lobato et al., 2011]. Weighted resampling with replacement is used to generate the bootstrap samples on which the individual classifiers are trained. In vote-boosting, the symmetric beta distribution is used for emphasis. The shape parameter is determined as the value among those in $a{=}b \in \{0.25, 0.5, 0.75, 1.0, 1.25, 1.5, 2.5, 5, 10, 20, 40\}$ that minimizes the 10-fold cross-validation in the training set. When uniform emphasis in all the training instances ($a = b = 1.0$) is used, the results are equivalent to bagging or, if random trees are used as base learners, to random forest. For values of the shape parameter above 1.0, the symmetric beta distribution has a single mode at $\pi = 0.5$, which implies that more emphasis is made on training instances on which the degree of disagreement among the

different classifiers is large. By contrast, when the shape parameter is smaller than 1 the focus is on training instances on which most classifiers agree.

For the empirical evaluation carried out in this and the following section, different binary classification tasks from the UCI repository [Bache and Lichman, 2013] and other sources [Breiman, 1998] are considered. The characteristics of the datasets used in this study are summarized in Table 5.1. For each dataset, the table displays the total number of labeled instances available, the number of those instances used for training and for testing, and the number of attributes. The test error rates reported are averages, followed by the corresponding standard deviations after the $\pm$ sign, over 100 realizations of the training and test sets. For classification problems in which only a finite collection of labeled instances is available, 2/3 of the data are selected at random for training and the remaining 1/3 for testing. For synthetic problems (namely, *Ringnorm*, *Twonorm*, and *Threenorm*), instances are generated independently at random: 300 instances are used for training and 2000 for testing. In all cases, stratified sampling is used to ensure that the class distributions in the test and training sets are similar.

In Table 5.2, the average test errors over the 100 realizations are shown for all base learners. For each dataset, the lowest average generalization error is highlighted in boldface. The second best result is underlined. If the differences between the two lowest test errors is statistically significant at a significance level $\alpha = 0.05$ the lowest error value is marked with an asterisk (*). A resampled paired t-test is used for synthetic problems. When random train/test partitions of the same dataset are employed a corrected resampled paired t-test [Bouckaert and Frank, 2004; Nadeau and Bengio, 2003] is used instead.

To provide an overall comparison of the accuracies of vote-boosting using the four tested base learners, we apply the framework proposed in [Demsar, 2006]. To this end, the average rank of the classifier average errors is computed for the 23 classification problems investigated. The rank of a classifier in a specific classification problem is determined by ordering the different methods according to their test errors. A lower rank corresponds to a smaller test error and, therefore, better accuracy. The average ranks of vote-boosting using the four different base learners are displayed in Figure 5.6. In this diagram, the differences of average ranks between methods that are connected by a horizontal solid line are not statistically significant according to a Nemenyi test (p-value < 0.05).

From the results presented in Table 5.2 and Figure 5.6, one concludes that vote-boosting composed of random trees has the best overall accuracy: except in *Boston*, *Horse-colic*, and *Parkinsons*, these types of ensembles achieve the lowest or second lowest average test errors. Notwithstanding, according to the Nemenyi test, the average rank differences are statistically significant only with respect to the use of stumps (see Figure 5.6).

Table 5.1
Characteristics of the classification problems analyzed and training / test partitions.

Dataset	**Instances**	**Training**	**Test**	**Number of attributes**
Adult	32561	21707	10854	15
Australian	690	460	230	14
Breast W.	699	466	233	9
Blood	748	499	249	5
Boston	506	337	169	14
Chess	3196	2131	1065	36
German	1000	667	333	20
Heart	270	178	92	13
Hepatitis	155	104	51	19
Horse-colic	368	246	122	21
Ionosphere	351	234	117	34
Liver	345	230	115	6
Magic	19020	12680	6340	11
Musk	6598	4399	2199	168
Ozone	2536	1691	845	74
Parkinsons	197	132	65	24
Pima	768	512	256	8
Ringnorm	2300	300	2000	20
Sonar	208	139	69	60
Spambase	4601	3068	1533	58
Threenorm	2300	300	2000	20
Tic-tac-toe	958	639	319	9
Twonorm	2300	300	2000	20

The values of the shape parameters of the symmetric beta distribution ($a = b$) selected by cross-validation are shown in Table 5.3. The figures reported are medians over the 100 realizations of the training and test data. An asterisk is shown in the Table when ensembles built using the different values have the same within-train cross-validation error. For decision stumps, this occurs in *Australian* and *Musk* because the individual stumps are the same irrespective of the type of emphasis employed. In *Musk*, when more complex base classifiers are used, the ensembles built with the different values of $a = b$ all achieve zero error. In most cases, the value of the shape parameter of the beta distribution ($a = b$) decreases as the complexity of the base classifier increases. The reason of this is that more emphasis is needed in order to *boost* more stable base classifiers. Thus, the largest values of $a = b$ are selected for decision stumps. Hence, for these types of base learners the focus is on uncertain instances, on which most ensemble classifiers disagree. The lowest values of $a = b$ correspond to random trees. In this case, the emphasis on uncertain instances is reduced. In fact, for *Blood*, *Heart*, *Musk*, and *Pima*, the focus should be on instances on which the ensemble classifiers agree.

Table 5.2
Test error rates of vote-boosting ensembles composed of decision stumps, pruned CART trees, unpruned CART trees and random trees.

Dataset	Stump	Pruned	Unpruned	Random tree
Adult	20.9±2.9	13.6±0.3	**13.5±6.2**	13.6±0.2
Australian	14.6±2.1	13.6±2.0	13.5±2.1	**13.3±2.1**
Breast W.	3.9±1.0	3.8±1.1	4.0±1.2	**3.1±1.1**
Blood	22.7±1.5	22.2±1.7	**21.8±1.8**	21.9±1.7
Boston	15.2±2.3	**12.7±2.5**	13.0±2.6	13.1±2.3
Chess	17.7±14.1	0.6± 0.6	0.7±0.6	**0.6±0.3**
German	25.4±1.6	24.1± 2.1	**24.1±2.0**	24.3±1.7
Heart	**16.4±3.5**	17.8±3.6	18.4±3.9	16.8±3.5
Hepatitis	16.4±4.3	16.2±4.8	16.9±4.9	**14.0±3.9**
Horse-Colic	**13.9±2.7**	14.1±2.9	13.9± 2.7	15.0±2.5
Ionosphere	8.1±1.7	6.7±2.0	6.7±1.8	**6.6±1.8**
Liver	**27.5±3.9**	28.4±3.6	28.6±3.9	28.4±3.7
Magic	15.4±0.3	12.7±0.3	12.9±0.3	**11.6±0.2***
Musk	**0.0±0.0**	**0.0±0.0**	**0.0±0.0**	**0.0±0.0**
Ozone	6.3±0.0	**5.8±0.4**	5.8±0.4	6.0±0.2
Parkinsons	11.5±4.3	**7.4±3.6**	7.6±3.2	9.3±3.6
Pima	24.1±2.0	24.2±2.2	23.8±2.3	**23.4±1.8**
Ringnorm	9.3±0.9	4.5±0.8	4.6±0.8	**4.4±0.8**
Sonar	18.9±4.6	16.0±4.4	15.8±4.8	**15.5±4.5**
Spambase	6.1±0.6	4.5±0.3	4.5±0.3	**4.1±0.5**
Threenorm	20.1±1.0	17.1±1.1	17.0±1.0	**16.2±1.0***
Tictactoe	22.0±2.9	**0.7±0.6**	0.7± 0.6	1.1±0.7
Twonorm	4.6±0.7	4.2±0.6	4.2±0.6	**3.6±0.5***

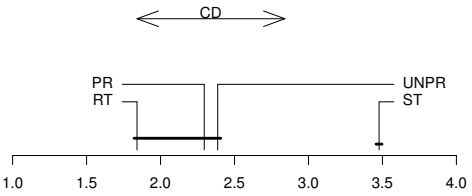


Fig. 5.6. Comparison of the average ranks of decision stumps (ST), pruned CART trees (PR), unpruned CART trees (UNPR), and random trees (RT) using a Nemenyi test. Horizontal lines connect methods whose average ranks are not significantly different (p-value < 0.05).

In summary, vote-boosting ensembles composed of random trees provide the best overall accuracy in the problems investigated. Since this type of ensemble can also be built efficiently, they will be used for further evaluation in the following set of experiments.

Table 5.3
Median of the beta distribution parameter ensembles composed of decision stumps, pruned CART trees, unpruned CART trees, and random trees. An asterisk is shown when all values of the beta parameter yield the same cross-validation error

Dataset	Stump	Pruned	Unpruned	Random tree
Adult	40.0	40.0	20.0	1.0
Australian	*	5.0	2.5	1.5
Breast W.	10.0	5.0	10.0	1.0
Blood	40.0	1.25	0.75	0.5
Boston	40.0	20.0	10.0	2.0
Chess	20.0	20.0	20.0	10.0
German	40.0	5.0	2.5	2.5
Heart	10.0	2.5	1.5	0.5
Hepatitis	10.0	10.0	5.0	2.5
Horse-Colic	20.0	1.5	1.5	1.5
Ionosphere	40.0	5.0	10.0	2.5
Liver	40.0	5.0	5.0	1.5
Magic	40.0	40.0	40.0	20.0
Musk	*	*	*	0.7
Ozone	40.0	10.0	5.0	1.5
Parkinsons	40.0	15.0	10.0	5.0
Pima	15.0	1.5	1.0	0.5
Ringnorm	40.0	20.0	20.0	20.0
Sonar	40.0	20.0	20.0	10.0
Spambase	40.0	20.0	20.0	20.0
Threenorm	40.0	10.0	10.0	5.0
Tictactoe	40.0	20.0	20.0	10.0
Twonorm	20.0	20.0	20.0	5.0

5.5.3 Comparison of vote-boosting with other ensemble methods

In this section, we carry out an comparison of vote-boosting ensembles and other related methods: bagging, AdaBoost and random forest. The classification problems considered and experimental protocol followed are the same as in the previous section. In each type of ensemble, the base classifier that performs best is used: pruned CART trees in AdaBoost, unpruned CART trees in bagging, and random trees in vote-boosting and random forest. Note that unpruned CART or random trees cannot be used in combination with AdaBoost because they generally achieve zero training error, which gives rise to singularities in the weight updates. As illustrated in the previous section, similar results are obtained with vote-boosting ensembles composed of unpruned or pruned CART trees. The adabag [Alfaro et al., 2013], ipred [Peters and Hothorn, 2009] and ranfomForest [Liaw and Wiener, 2002b] packages in R have been used for AdaBoost, bagging, and random forest implementations, respectively. As in the previous subsection, ensembles of 501 classifiers are used to guarantee convergence to the asymptotic error level. In both vote-boosting and AdaBoost, weighted resampling with replacement is

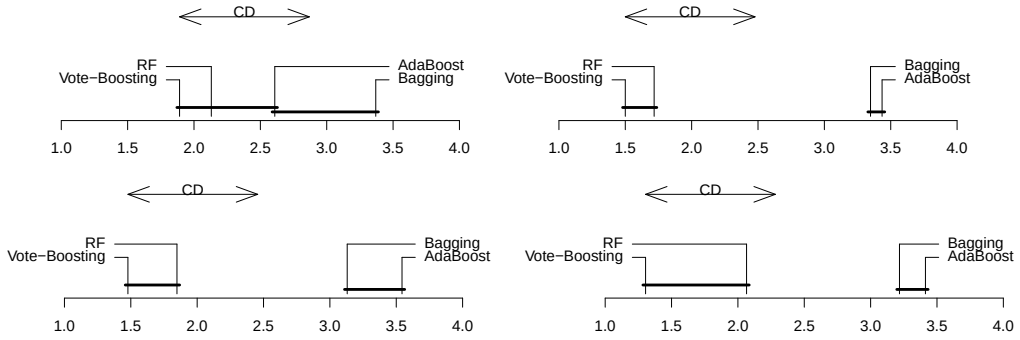


Fig. 5.7. Comparison of the average ranks of bagging, AdaBoost, random forest and vote-boosting using a Nemenyi test. Horizontal lines connect methods whose average ranks are not significantly different (p-value < 0.05). The plots correspond to datasets without injected noise (top left), with 10% (top right), 20% (bottom left), and 30% class-label noise (bottom right).

used to take into account the different emphasis on the training instances. In AdaBoost, reweighting was considered as an alternative. However, as reported in the literature, similar or slightly better results are obtained when resampling instead of reweighting is used [Dietterich, 2000a; Seiffert et al., 2008].

The experiments are carried out on the original problem, and also with 10%, 20% and 30% class-label noise. Class-label noise is injected into the training set by randomly switching the class label of a random subset (10%, 20% and 30%) of the training instances. This type of label noise is know as completely at random noise (NCAR) [Frénay and Verleysen, 2014]. An interesting observation is that, as the noise level increases, the values of the shape parameter that are selected tend to decrease.

The test error rates of the different ensembles and classification problems considered are displayed in Tables 5.4, 5.5, and 5.6. The results reported are averages, followed by the corresponding standard deviations after the $\pm$ sign, over 100 realizations of the training and test set partitions. The median value for $a = b$ used in vote-boosting is reported in the last columns of Tables 5.4, 5.5, and 5.6. In these tables, the best and second best results for each classification problem are highlighted using bold face and underlined, respectively. In addition, the lowest test error rate is marked with an asterisk (*) if the improvement over to the second best is statistically significant, at a significance level $\alpha = 0.05$. The significance of these differences is determined using a paired resampled t-test for synthetic problems, and to a corrected resampled paired t-test [Bouckaert and Frank, 2004; Nadeau and Bengio, 2003] when random train/test partitions are used. Finally, the number of significant wins and losses when one compares the average accuracy of vote-boosting with each of the remaining ensembles, according to the aforementioned statistical test, are shown in the last row of Table 5.6. Draws correspond to differences of average accuracy that are not statistically significant.

Table 5.4
Test error rates for bagging (unpruned CART trees), boosting (pruned CART trees), random forest, and vote-boosting (random trees).

Dataset	Noise (in %)	Bagging	AdaBoost	Random forest	Vote-boosting	a=b (median)
Adult	0	14.6±0.2	13.8±0.2	13.6± 0.2	**13.6±0.2**	1.0
	10	15.5±0.3	14.3±0.3	14.1±0.3	**14.0±0.3**	0.75
	20	17.0±0.3	14.8±0.2	14.8±0.3	**14.3±0.2**	0.25
	30	20.3±0.3	15.1± 0.3	17.8±0.3	**15.1±0.2**	0.25
Australian	0	13.3±2.1	13.7±1.9	**13.0±2.0**	13.3±2.1	1.5
	10	14.9±2.3	17.6±2.4	**13.6±2.0**	13.6± 2.1	0.75
	20	18.2±2.9	24.3±3.0	15.8±2.2	**14.3±2.4***	0.25
	30	24.6±3.8	32.0±3.5	21.6±3.0	**18.2±3.3***	0.25
Breast W.	0	4.6±1.2	3.6±1.0	**3.0±1.0**	3.1±1.1	1.0
	10	6.3±1.6	6.9±1.8	4.1±1.2	**3.5±1.2***	0.5
	20	9.6±2.5	12.0±2.6	6.5±1.9	**4.1±1.4***	0.25
	30	17.7±3.0	20.9±3.4	13.0±2.7	**6.8±2.6***	0.25
Blood	0	26.3±1.9	25.4±2.1	24.5±2.2	**21.9±1.7***	0.5
	10	28.5±2.2	27.2±2.5	26.6±2.4	**22.6±2.0***	0.25
	20	31.6±2.7	30.2±2.6	29.7±2.6	**23.8±2.5***	0.25
	30	35.8±3.6	34.6±3.8	34.2±3.6	**29.0±4.5***	0.25
Boston	0	14.2±2.4	**12.7±2.4**	13.0±2.4	13.1±2.3	2.0
	10	16.2±2.8	16.9±2.5	**14.5±2.6**	14.9±2.5	0.5
	20	19.1±3.3	22.2±3.6	17.5±3.2	**16.2±2.8**	0.5
	30	26.4±4.2	31.4±4.0	24.6±3.8	**21.3±4.4***	0.25
Chess	0	0.6±0.3	**0.5±0.3**	1.6±0.4	0.6±0.3	10.0
	10	5.1±0.7	2.3±0.5	2.1± 0.5	**2.1±0.5**	1.25
	20	11.9±1.2	**3.6±0.7**	4.3±0.9	4.0±0.9	0.75
	30	22.4±1.5	**5.5±0.9***	10.2±1.3	7.9±1.5	0.25
German	0	24.5±2.0	24.8±2.1	**24.0±1.8**	24.3±1.7	2.5
	10	26.2±2.1	27.9±2.1	25.3± 1.8	**25.3±1.9**	1.0
	20	28.8±2.4	32.2±2.5	**27.3±2.2**	27.4±2.3	0.75
	30	32.3±3.0	37.4±2.8	31.0±3.1	**29.9±3.2**	0.5
Heart	0	19.5±3.7	20.2±3.4	17.2±3.0	**16.8±3.5**	0.5
	10	22.7±4.2	24.9±4.5	19.3±3.7	**18.7±4.0**	0.5
	20	26.0±5.0	30.6±5.1	22.7±4.2	**20.8±4.0**	0.375
	30	32.1±5.4	36.3±6.0	28.5±5.8	**25.8±6.6**	0.25
Hepatitis	0	15.8±4.7	15.8±4.0	**13.6±3.5**	14.0±3.9	2.5
	10	17.1±4.7	20.0±5.0	**14.2±3.9**	15.3±3.9	1.0
	20	21.0±5.8	26.0±6.9	17.6±4.8	**17.3±4.5**	0.5
	30	27.8±7.8	33.6±7.2	23.8±7.3	**21.9±7.0**	0.5
Horse-colic	0	**14.9±2.8**	15.4±2.6	15.0±2.5	15.0±2.5	1.5
	10	17.8±3.5	21.2±3.5	**17.3±2.9**	17.4±3.0	1.0
	20	23.4±4.2	27.6±4.7	21.3±3.6	**20.9±3.7**	0.625
	30	30.0±5.6	34.9±5.1	28.4±4.6	**28.2±5.3**	0.75

Table 5.5
Test error rates for bagging (unpruned CART trees), boosting (pruned CART trees), random forest, and vote-boosting (random trees).

Dataset	Noise (in %)	Bagging	AdaBoost	Random forest	Vote-boosting	a=b (median)
Ionosphere	0	8.0±2.2	**6.6±1.8**	6.6±1.6	6.6± 1.8	2.5
	10	9.5±2.7	10.5±2.6	7.9± 2.2	**7.9±2.2**	0.75
	20	13.1±3.1	17.0±3.5	10.9±2.9	**9.9±3.1**	0.5
	30	19.7±4.5	26.4±4.5	17.8±4.6	**15.7±5.1**	0.25
Liver	0	29.5±3.9	30.5±3.9	**27.7±3.8**	28.4±3.7	1.5
	10	31.7±4.1	33.8±4.5	**31.0±4.0**	31.4±3.8	1.0
	20	36.1±4.3	37.8±4.5	**35.1±4.4**	35.7±4.4	1.0
	30	39.9±4.7	41.5±4.7	**39.6±4.5**	39.7±4.9	0.875
Magic	0	12.3±0.3	12.9±0.3	12.0±0.3	**11.6±0.2***	20.0
	10	12.9±0.3	13.9±0.3	**12.4±0.2**	12.5±0.2	1.25
	20	14.2±0.4	14.4±0.3	13.6±0.4	**13.3±0.4***	0.5
	30	17.4±0.5	15.0±0.5	16.7±0.4	**14.6±0.4***	0.25
Musk	0	**0.0±0.0**	**0.0±0.0**	**0.0±0.0**	**0.0±0.0**	0.7
	10	1.4±0.3	0.6±0.2	1.6±0.3	**0.1±0.1***	0.25
	20	4.5±0.4	0.9± 0.6	5.7±0.4	**0.8±0.3**	0.25
	30	12.0±1.0	**2.5±1.2***	14.4±1.2	5.1±0.7	0.25
Ozone	0	6.0±0.3	**5.7±0.4***	6.0±0.2	6.0±0.2	1.5
	10	6.2±0.3	6.4±0.5	6.0± 0.2	**6.0±0.3**	5.0
	20	6.6±0.6	9.7±1.0	**6.1±0.4**	6.2±0.4	1.0
	30	9.1±0.9	18.3±1.5	7.9±1.0	**7.5±1.2**	0.5
Parkinsons	0	10.5±4.0	**7.1±3.3***	10.2±3.4	9.3±3.6	5.0
	10	13.8±4.2	12.9±4.0	**12.2±3.7**	12.7±4.2	1.5
	20	18.6±5.7	20.1±5.9	**16.2±4.8**	17.0±4.9	0.75
	30	24.1±5.5	28.0±6.0	23.0±5.8	**22.6±6.1**	0.5
Pima	0	23.9±1.9	25.9±1.9	**23.2±2.0**	23.4±1.8	0.5
	10	25.9±2.3	28.9±2.3	24.7±2.2	**24.1±2.4**	0.25
	20	28.4±2.6	32.9±3.3	26.7±2.5	**25.3±2.5***	0.25
	30	32.6±3.0	38.4±3.3	31.5±2.8	**29.8±3.7***	0.5
Ringnorm	0	8.9±1.8	**4.3±0.6***	6.0±1.1	4.4±0.8	20.0
	10	9.6±1.7	7.6±1.0	6.7±1.2	**6.1±1.4***	10.0
	20	10.8±1.8	13.0±1.7	**7.9±1.4***	8.4±1.8	1.25
	30	15.6±2.9	22.2±2.3	**12.4±2.4**	12.5±3.0	0.75
Sonar	0	22.4±5.1	**15.1±4.6**	17.9±4.8	15.5±4.5	10.0
	10	23.3±5.5	20.7±5.1	20.6±5.4	**19.7±4.6**	5.0
	20	26.9±5.7	26.3±5.3	**24.2±5.7**	24.5±5.6	1.25
	30	32.2±5.0	34.6±5.5	**30.4±5.4**	30.4± 5.3	0.75
Spambase	0	5.9±0.5	4.3±0.4	5.0±0.5	**4.1±0.5**	20.0
	10	8.2±0.8	**6.1±0.6**	6.5±0.6	6.3±0.6	0.75
	20	11.5±1.0	7.5±0.7	9.4±0.8	**7.1±0.7**	0.25
	30	16.5±1.1	10.3±1.1	14.3±1.0	**8.6±0.8***	0.25

Table 5.6
Test error rates for bagging (unpruned CART trees), boosting (pruned CART trees), random forest, and vote-boosting (random trees).

Dataset	Noise (in %)	Bagging	AdaBoost	Random forest	Vote-boosting	a=b (median)
Threenorm	0	19.0±1.7	16.8±0.8	16.6±0.9	**16.2±1.0***	5.0
	10	20.6±1.7	20.2±1.2	**18.5±1.1**	18.6±1.2	1.5
	20	23.3±1.8	24.9±1.6	**21.2±1.3***	21.6±1.5	1.25
	30	28.8±2.1	32.0±1.9	**26.9±2.0***	27.2±2.5	0.62
Tic-tac-toe	0	2.1±0.9	**0.7±0.6***	2.1±1.0	1.1±0.7	10.0
	10	5.7±1.6	9.9±1.8	5.9±1.6	**5.6±1.6**	2.5
	20	**12.5±2.2**	20.6±2.7	12.7±2.6	13.0±2.6	1.25
	30	23.5±2.8	30.3±2.9	**22.2±2.7**	22.9±2.9	0.75
Twonorm	0	6.4±1.5	4.0±0.5	3.9±0.5	**3.6±0.5***	5.0
	10	7.3±1.6	7.1±0.9	**4.8±0.7***	4.9±0.8	1.25
	20	9.3±2.1	12.6±1.6	**6.6±1.1**	6.7±1.2	0.75
	30	14.2±2.3	21.5±2.4	10.7±1.7	**9.6±2.5***	0.5
win/draw/loss	0	12/11/0	6/13/4	9/14/0	—	
	10	16/7/0	17/6/0	4/18/1	—	
	20	18/5/0	16/7/0	7/14/2	—	
	30	19/4/0	19/2/2	11/11/1	—	

From the results presented in these tables, it is clear that vote-boosting ensembles exhibit the best overall performance. In particular, it has the largest number of statistically significant wins at all noise levels. The tally is very favorable when one compares the average accuracy of vote-boosting with bagging: vote-boosting significantly outperforms bagging in 12 out of the 23 datasets (without injected noise). For 10%, 20% and 30% noise levels, the differences in number of wins become larger: vote-boosting significantly outperforms bagging in 16, 18 and 19 out of the 23 tested datasets, respectively. The comparison with AdaBoost on the explored datasets is also favorable to vote-boosting. In the original datasets (i.e.without injected noise) vote-boosting outperforms AdaBoost in 6 out of 23 datasets and is inferior in 4 datasets: *Ozone*, *Parkinsons*, *Ringnorm*, and *Tic-tac-toe*). In this problems vote-boosting is second best; furthermore, the test error rates of vote-boosting are fairly close to AdaBoost. In these classification problems, except for *Ozone*, the shape parameter of the beta distribution used as emphasis function in vote-boosting is fairly high. This indicates a strong emphasis on uncertain examples, which has a similar effect as the emphasis on incorrectly classified instances that is characteristic of AdaBoost. On the other hand, in problems such as *Blood*, *Heart*, *Liver* or *Pima*, which are difficult for AdaBoost, vote-boosting selects low values for $a = b$, which implies that less emphasis is made on uncertain examples.

It is remarkable that, for some datasets, such as *Blood*, *Heart*, and *Pima*, and in most of the problems, for sufficiently high levels of class-label noise, values of $a = b$ below 1.0 provide the best accuracy. In such cases, emphasis is made on instances on which the individual ensemble classifiers agree the most. The effectiveness of this type of emphasis,

which is somewhat counter-intuitive, is a consequence of the large intrinsic variability of random trees. In fact, this effect is less prominent in ensembles of more stable base learners, such as decision stumps or pruned CART trees. As the noise level increases, the performance of AdaBoost rapidly deteriorates. By contrast vote-boosting is robust to class-label noise: it outperforms AdaBoost in 17, 16 and 19 datasets for 10%, 20% and 30% injected noise, respectively.

Finally, vote-boosting is more accurate than random forest in 9 out of the 23 classification problems investigated for the noiseless case. In some cases, the improvements can be fairly large (e.g. *Blood*, *Chess*, *Ringnorm*, *Sonar* or *Tic-tac-toe*). In 14 datasets the two methods have comparable accuracies. The accuracy improvements of random forests over vote-boosting are not statistically significant in any of the problems investigated. When the class labels are contaminated, vote-boosting performs significantly better in 4, 7 and 11, and worse in 1, 2 and 1 datasets for 10%, 20%, and 30% noise levels, respectively. Again, in the noisy datasets we see that, when random forest wins, the differences are typically small. By contrast, when vote-boosting wins, the differences are, in general, large.

In addition, the overall performance of the accuracies of the different ensembles are summarized in Figure 5.7 using the procedure described in [Demsar, 2006]. The plots in this figure display the average ranks of bagging, AdaBoost, random forest (RF) and vote-boosting for the original datasets (top left), and for 10% (top right), 20% (bottom left), 30% (bottom right) injected class-label noise. In these plots, a horizontal solid line connects methods for which the differences of average ranks are not statistically significant using a Nemenyi test with p-value < 0.05. In the original problems, vote-boosting has the best average rank. However, the differences with random forest and AdaBoost are not statistically significant. The difference with bagging, which has the worst performance in terms average rank, is statistically significant. For problems contaminated with noise in the class labels, vote-boosting has the best average rank. The differences with respect to random forest increase for higher noise levels. However, they are not statistically significant. In all noisy problems, the average ranks of both random forest and vote-boosting ensembles are significantly better than bagging and AdaBoost.

5.5.4 Emphasis profiles

From the values of the shape parameter of the beta distribution reported in the last column of Tables 5.4, 5.5, and 5.6, it is apparent that the type of emphasis and its strength need to be adapted to the classification task considered. To further investigate

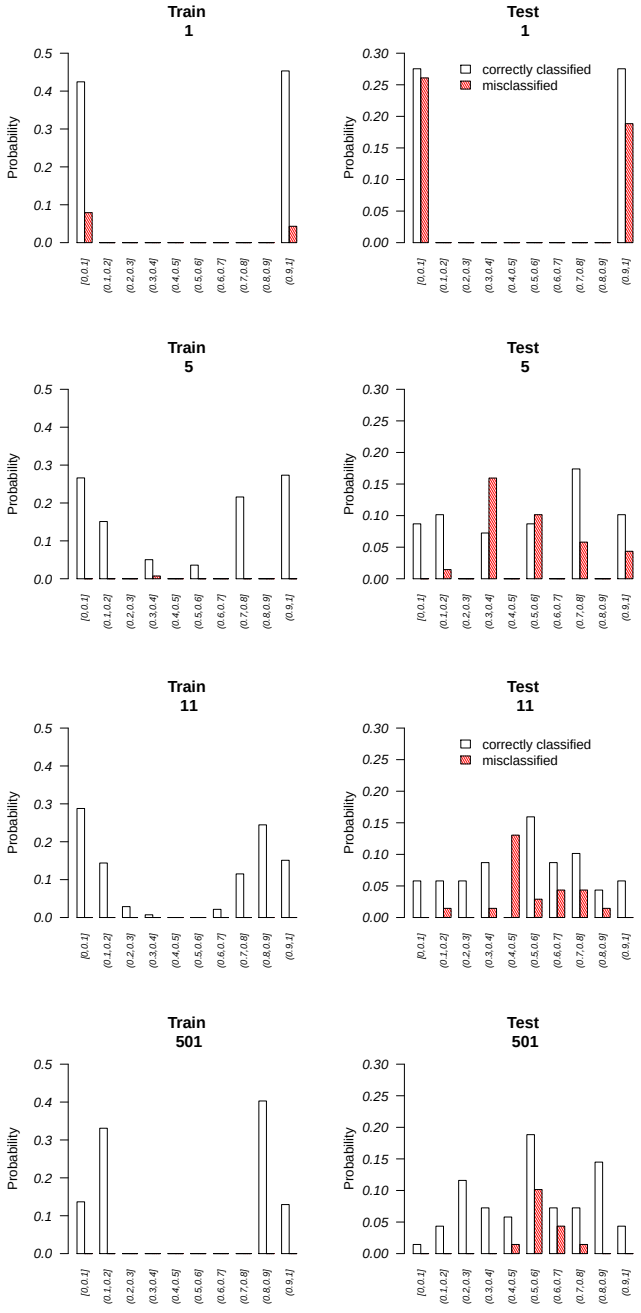


Fig. 5.8. Histograms of vote fractions for correctly (white) and incorrectly (red) classified instances in *Sonar* for the training set (left column) and test set (right column)

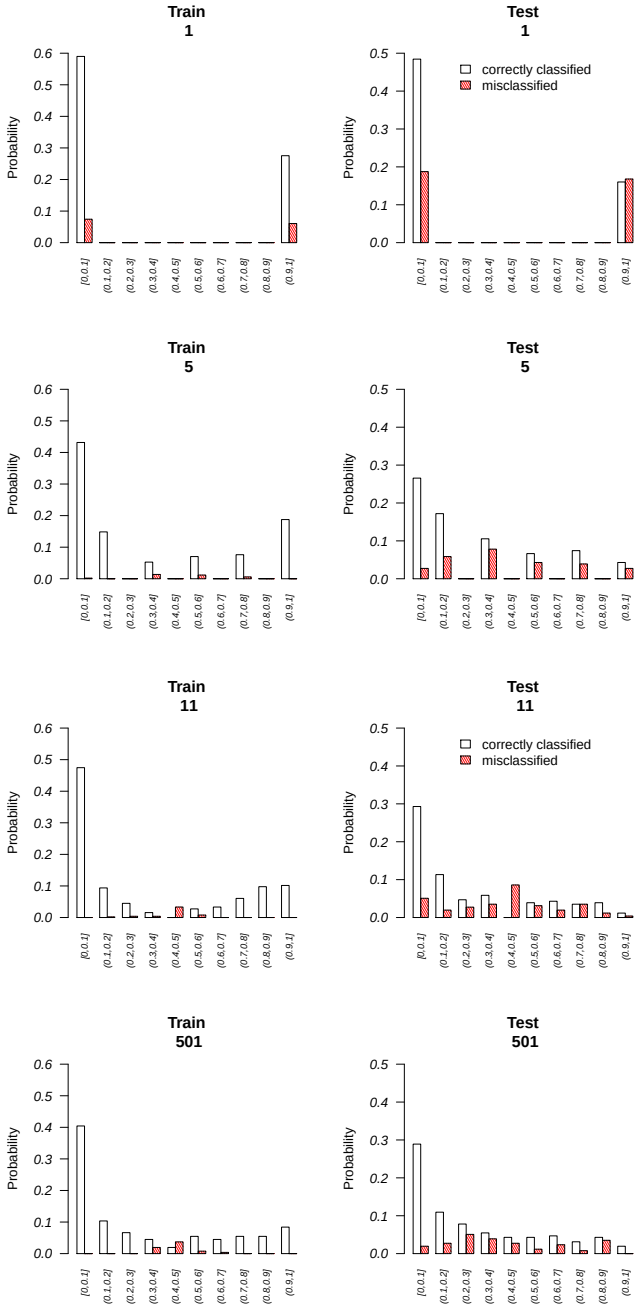


Fig. 5.9. Histograms of vote fractions for correctly (white) and incorrectly (red) classified instances in *Pima* for the training set (left column) and test set (right column)

this point, we have carried out a detailed analysis of the distribution of class votes for *Sonar* and *Pima*, in which the optimal emphasis strategies are very different.

In *Sonar*, the vote-boosting ensemble is built using a beta distribution with $a = b = 10.0$. Thus, the optimal emphasis is to focus on training instances in which the disagreement rates are largest. In Figure 5.8, the histograms of the distribution of votes are plotted for ensembles of 1, 5, 11 and 501 random trees. The height of the white bars indicate the fraction of correctly classified instances for the corresponding range of voting distributions. The red stripped bars correspond to incorrectly classified instances. The plots on the left are for the training set and on the right for the test set. In the training set, the strong focus on uncertain instances (those for which the fraction of class votes is close to 0.5) leads to a markedly bimodal distribution, in which most predictions are by clear majority. Incorrectly classified instances disappear because ensembles that are sufficiently large achieve zero training error. The distribution of class votes in the test set is markedly different: It covers the whole interval, and exhibits a low peak for intermediate class vote frequencies, especially for instances that are misclassified.

A very different picture is obtained in *Pima* (Figure 5.9). In this classification task, the selected shape parameter for the symmetric beta distribution is $a = b = 0.5$. In consequence, the optimal strategy is to avoid focusing on training instances in which the disagreement rates are large. For correctly classified instances, the histograms in training and test sets are similar. Misclassified instances in the training set appear mostly around 0.5. By contrast, in the test set, they appear in the whole $[0, 1]$ interval. This is consistent with the observation that *Pima* has high levels of class-label noise.

5.6 Conclusions

Vote-boosting is a novel ensemble learning method in which individual classifiers are built using different weighted versions of the training data. To build a new classifier, the weights of the training instances are determined in terms of the disagreement rate among the classifiers that make up the ensemble. The optimal weighing scheme depends on the complexity of the base classifiers and on the level of noise in the class labels of the training data. For simple or regularized classifiers, such as decision stumps or pruned CART trees, vote-boosting interpolates between bagging and AdaBoost. When the level of class-label noise is small, prediction errors are more likely to occur near the classification boundary, where the uncertainty, as measured by the disagreement among ensemble predictions, is largest. Therefore, it is possible to build more accurate ensembles by focusing on uncertain instances. Since these instances are more likely to be misclassified, the emphasis given by vote-boosting is similar to AdaBoost's. For noisy

classification problems, a softer emphasis on uncertain instances is generally preferable. In this case, the most accurate predictions are obtained by means of ensembles that are fairly similar to bagging. When more variable individual learners are used (e.g. unpruned CART or random trees) a milder emphasis on uncertain instances is generally needed to achieve the best generalization performance. This is a consequence of the fact that some of the uncertainty in the predictions is due to the intrinsic variability of the base learners. For problems in which the level of class-label noise is high it is in fact advantageous to progressively focus on instances in which the ensemble classifiers agree. In practice, the optimal type of emphasis can be readily determined using cross-validation within the training data.

Note that, since AdaBoost is based on emphasizing incorrectly classified instances, it cannot be used to improve the performance of base learners whose training error is small, such as unpruned CART or random trees. By contrast, vote-boosting does not have this limitation and can be used to build boosted ensembles composed of these types of classifiers that are both accurate and robust to noise in the class labels.

Acknowledgements

The research has been supported by the Spanish *Ministry of Economy, Industry, and Competitiveness* (projects TIN2016-76406-P, TIN2013-42351-P and TIN2015-70308-REDT), and *Comunidad de Madrid*, project CASI-CAM-CM (S2013/ICE-2845).

Chapter 6

Randomization vs optimization in SVM ensembles

6.1 Abstract

Ensembles of SVMs are notoriously difficult to build because of the stability of the model provided by a single SVM. The application of standard bagging or boosting algorithms generally leads to small accuracy improvements at a computational cost that increases with the size of the ensemble. In this work, we leverage on subsampling and the diversification of hyperparameters through optimization and randomization to build SVM ensembles at a much lower computational cost than training a single SVM on the same data. Furthermore, the accuracy of these ensembles is comparable to a single SVM and to a fully optimized SVM ensemble.

6.2 Introduction

The SVM algorithm has received much attention in the machine learning community because of its strong theoretical foundations and its state-of-the-art performance in a wide range of applications [Burges, 1998; Cortes and Vapnik, 1995b]. In a binary classification problem, an SVM is built by finding the maximum-margin hyperplane that separates the two classes. The parameters of the hyperplane are determined by solving a convex optimization problem that can be formulated in terms of scalar products. To allow for the possibility of class overlap, a regularization term that penalizes errors in the training set and preserves the convexity of the optimization problem is included in the objective function. The strength of this regularization is quantified by a non-negative

constant C whose value needs to be carefully adjusted. Finally, a non-linear classifier that maximizes the margin can be built by replacing the scalar products that appear in the objective function of the optimization problem by the corresponding inner products in a Reproducing Hilbert Space associated to a kernel. Linear, polynomial, or RBF kernels are typically used for this embedding. In practice, SVMs built with an RBF kernel have good generalization capacity provided that the value of the kernel width $(1/\gamma)$ is properly adjusted [Cherkassky and Ma, 2004].

In spite of their success, there are some difficulties in the practical application of SVMs. The main one is the high computational cost of training. This disadvantage is exacerbated by their sensitivity to the values of the hyperparameters (C,γ), which are commonly selected by grid search using a costly cross-validation procedure. A possible way to improve the performance of a single SVM is to build ensembles [Claesen et al., 2014; Kim et al., 2003; Mayhua-López et al., 2015; Stork et al., 2015; Valentini and Dietterich, 2004; Vapnik, 1999]. If subsampling is used to train the individual SVMs, building an ensemble can be faster than training a single SVM [Claesen et al., 2014] on the same data. In general, the improvements of an ensemble over a single SVMs are generally small. The reason is that SVM are strong and stable classifiers. In consequence, they are difficult to diversify without introducing large distortions that reduce their accuracy. The goal of this work is to design ensembles of SVMs that are at least as accurate as a single SVM at a reduced computational cost. To this end, we leverage on subsampling and explore the interplay between optimization and randomization methods in the determination of the hyperparameters of the individual SVMs in the ensemble.

6.3 SVM ensembles

In this work we analyze three strategies to build bootstrapped ensembles of SVMs: The completely-optimized SVM ensemble (COSE), the partially-optimized SVM ensemble (POSE), and the randomized-optimized SVM ensemble (ROSE). The individual SVMs in all these ensembles are built using independent bootstrap samples drawn from the original training data. The strategies differ in the way that the hyperparameters for the individual SVMs are chosen. In the completely-optimized SVM ensemble (COSE), optimal values of C and γ are selected for each individual SVM in the ensemble. In the partially-optimized SVM ensemble (POSE), optimal combinations of the SVM hyperparameters $\{(C_b, \gamma_b)\}_{b=1}^{B}$ are determined for $B = T/M \ll T$ different bootstrap samples of the original training data, where T is the desired size of the complete ensemble. The final ensemble is built in B batches. For each of these batches ($b = 1, \ldots, B$),

we fix the hyperparameters (C_b, γ_b) and build M different SVMs on independent bootstrap samples of the training data. Finally, in the randomized-optimized SVM ensemble (ROSE), T SVM's are are built on independent bootstrap samples using randomized values of the hyperparameters C and γ. From these, we select the best $\{(C_b, \gamma_b)\}_{b=1}^{B}$, for $B = T/M \ll T$. The final ensemble is built in B batches, in the same way as the POSE ensemble.

6.4 Empirical evaluation

We now present the results of an empirical evaluation of the strategies to build SVM ensembles introduced in the previous section. Specifically, the accuracy and training costs of the proposed methods are compared with those of SVM in 8 binary classification problems from the UCI repository [Bache and Lichman, 2013] and two synthetic ones (*Threenorm* and *Twonorm*). For the UCI problems, stratified random train/test partitions are used. The training set is composed of 2/3 of the labeled instances available for learning. The remaining 1/3 are set aside for testing. In the synthetic classification problems we generate 300 examples for training and 2000 for testing. The attributes of the instances are normalized so that they have zero mean and unit variance in the training set. The results reported are averages over 10 realizations of the classification problems: either random partitions for real-world data, or independent generations of the training/test sets for synthetic data. The methods are implemented in Python using Scikit-learn library [Pedregosa et al., 2011].

We have considered the use of bootstrap samples built either with replacement, as in standard bagging, or without replacement, as in subbagging [Bühlmann and Yu, 2002; Friedman and Hall, 2007]. The size of the bootstrap samples with replacement coincides with the original training data. The size of the bootstrap samples without replacement is 50 % of the original one. Subbagging using this ratio is expected to yield similar results as standard bagging [Friedman and Hall, 2007; Martínez-Muñoz and Suárez, 2010]. In the classification problems considered, the overall accuracy of subbagging is slightly better than bagging. Even though the differences in accuracy are not statistical significant, there is a marked computational advantage of using subbaging. For this reason, only the results of subbagging are reported.

In all three ensemble methods considered and in the single SVMs, an RBF kernel is used. The values of the hyperparameters are selected from a grid in which $C = 2^q$ with $q = -5, \ldots, 15$ and $\boldsymbol{\gamma} = 2^p$ with $p = -15, \ldots, 3$. For the single SVM and for the individual SVMs in COSE, 10-fold cross-validation within the corresponding training sets is used to select the optimal values of these hyperparameters. In POSE and ROSE,

Table 6.1
Generalization error for a single SVM and for SVM ensembles

Dataset	**SVM**	**COSE**	**POSE**	**ROSE**
Australian	15.70±2.32	**14.09±1.62**	14.17±0.97	14.22±2.62
Boston	13.85±2.93	13.73±1.93	**13.43±2.14**	14.26±1.56
Colic	20.74±1.64	20.57±2.55	20.66±2.07	**20.41±2.43**
German	23.96±1.56	23.51±1.54	23.75±1.46	**23.48±1.63**
Heart	17.78±3.78	17.22±2.58	**16.78±2.69**	19.44±3.24
Parkinsons	8.62±2.73	9.85±3.26	10.77±3.40	**8.31±3.18**
Pima	23.28±2.24	22.15±2.16	**21.95±2.60**	22.23±2.74
Spambase	**6.30±0.65**	6.38±0.28	6.36±0.34	6.51±0.24
Threenorm	14.01±0.63	**13.51±0.56**	13.74±0.64	13.90±0.73
Twonorm	2.73±0.64	**2.46±0.15**	2.53±0.17	2.63±0.21

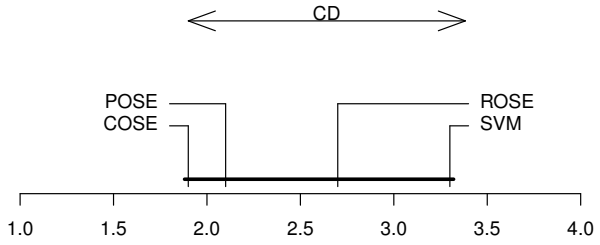


Fig. 6.1. Average ranks for SVM, COSE, POSE and ROSE (more details in the text)

$B = 10$ different pairs of hyperparameter values, $\{(C_b, \gamma_b)\}_{b=1}^{10}$, are selected using out-of-bag data. According to the empirical investigation carried out, the behavior of POSE and ROSE ensembles is not particularly sensitive to this parameter: ensembles built using values of B between 5 and 50 exhibit similar accuracies in the problems considered. The size of the ensembles generated is $T = 501$, which is sufficiently large for convergence of the classification errors to their asymptotic (optimal) limit [Hernández-Lobato et al., 2011].

A summary of the results of the experiments performed is given in table 6.1. For each dataset the errors rate of a single SVM, the completely-optimized (COSE), the partially-optimized (POSE), and the randomized-optimized (ROSE) SVM ensemble are shown. The values displayed are averages over 10 realizations of the classification problems considered, followed by the corresponding standard deviations after the ± sign. For each dataset, the most accurate method is highlighted in boldface. The second best is underlined. In addition, we have used the methodology proposed in [Demsar, 2006] to perform an overall comparison of the classifiers' performance across the different

datasets. Following this methodology, in Figure 6.1 the average ranks of the different methods are displayed. In this diagram, the differences in accuracy of methods that are connected by a horizontal solid line, are not statistically significant according to a Nemenyi test (p-value < 0.05). Finally, in Table 6.2 we report the training time in seconds to build a single SVM with grid search (second column) and speed-up obtained in training for each method relative to the SVM training time (last three columns). These times were measured on a single core of a CPU Intel Core i5, 64 bits, 2.30 GHz with 8 GB of memory.

From Table 6.1 one observes that the single SVM is the most accurate predictor only in one dataset (*Spambase*). By contrast, each of the ensemble methods considered achieves the highest accuracy in three problems. The best overall accuracy, in terms of average ranks, correspond to COSE. However, the computational cost of COSE is enormous: around 50-100 times slower to train than a single SVM. Even though the differences are not statistically significant, the average rank of POSE and ROSE ensembles are higher than single SVMs. As shown by the speedup factors displayed in Table 6.2, these improvements in accuracy are achieved with much lower training costs: POSE ensembles are between 2 and 3 times faster to build than a single SVM. More impressively, the speedup factors for ROSE are between 5 and 20. Furthermore, the differences between the average ranks between each of these and COSE, which is the best ensemble according to this measure, are not statistically significant.

Table 6.2
Training times in seconds for a single SVM and speedup factors for COSE, POSE and ROSE with respect to SVM

Dataset	**SVM (s.)**	**COSE**	**POSE**	**ROSE**
Australian	63.9	1.8e-02	2.7	15.4
Boston	23.8	9.7e-03	2.3	9.6
Colic	23.2	9.7e-03	2.3	9.6
German	186.9	2.7e-02	2.8	18.4
Heart	10.8	5.6e-03	2.2	6.9
Parkinsons	7.3	4.1e-03	2.2	5.4
Pima	127.8	2.7e-02	3.3	19.6
Spambase	2892.4	2.7e-02	2.0	14.4
Threenorm	30.2	1.1e-02	2.4	10.6
Twonorm	18.8	8.4e-03	2.1	8.5
Average		1.5e-02	2.4	11.8
Stdev		9.1e-03	0.4	4.8

6.5 Conclusions

In this work, we have proposed and analyzed three types of fast and accurate SVM ensembles built using subagging. Each individual SVM is induced from a bootstrap sample that includes 50% of the original instances, without repetitions. The behavior of a subagging ensemble with this sample size is expected to be similar to the corresponding standard bagging ensemble. Different combinations of optimization and randomization are used to determine the hyperparameters of the individual SVMs in the ensemble: In COSE, the strength of the regularization term (C) and the inverse width of the RBF kernel (γ) are fully optimized. In POSE and ROSE, a small number of different values of these parameters $\{(C_b, \gamma_b)\}_{b=1}^{B}$ is used repeatedly to build individual SVMs. Specifically, each of these combinations of values is used $M = T/B$ times to build an ensemble of size T. In POSE the combinations of values are determined by optimization in B independent bootstrap samples. In ROSE, the best B out of T randomly generated combinations is selected. For ensembles of size $T = 501$, values of B between 5 and 50 lead to very accurate POSE and ROSE ensembles whose accuracy is comparable to the completely optimized ensemble (COSE) and slightly better than a single SVM. In addition, the training speed of POSE and ROSE is over 2 and 10 times faster than the training time of a SVM optimized using a standard grid search procedure.

Chapter 7

Pooling homogeneous ensembles to build heterogeneous ensembles

7.1 Abstract

In ensemble methods, the outputs of a collection of diverse classifiers are combined in the expectation that the global prediction be more accurate than the individual ones. Heterogeneous ensembles consist of predictors of different types, which are likely to have different biases. If these biases are complementary, the combination of their decisions is beneficial. In this work, a family of heterogeneous ensembles is built by pooling classifiers from M homogeneous ensembles of different types of size T. Depending on the fraction of base classifiers of each type, a particular heterogeneous combination in this family is represented by a point in a regular simplex in M dimensions. The M vertices of this simplex represent the different homogeneous ensembles. A displacement away from one of these vertices effects a smooth transformation of the corresponding homogeneous ensemble into a heterogeneous one. The optimal composition of such heterogeneous ensemble can be determined using cross-validation or, if bootstrap samples are used to build the individual classifiers, out-of-bag data. An empirical analysis of such combinations of bootstraped ensembles composed of neural networks, SVMs, and random trees (i.e. from a standard random forest) illustrates the gains that can be achieved by this heterogeneous ensemble creation method.

7.2 Introduction

Building an effective classifier for a specific problem is a difficult task. To be successful, a variety of aspects need to be taken into account: the structure of the data, the

information that can be used for prediction, the number of the labeled examples available for induction, the noise level, among others. Another crucial choice is the type of predictor to be used. The strategies implemented by the different classifiers are diverse. For instance, decision trees adopt a divide-and-conquer approach in which the original prediction task is recursively divided by partitioning the attribute space into disjoint regions. Within each of these regions, the prediction problem is simpler than the original. A neural network provides a global sub-symbolic representation of the decision problem in terms of the set of synaptic weights. Another illustration is the strategy adopted in kernel methods, such as Suppor Vector Machines (SVM). In SVMs the original problem is embedded into an extended feature space. In this extended space, the discrimination problem is solved by finding the maximized margin hyperplane that separates classes, except for, possibly, a few instances. In practice, one often finds that combining the outputs of individual classifiers often leads to more accurate predictions. Whence, the popularity of ensemble methods [Banfield et al., 2007; Bauer and Kohavi, 1999; Dietterich, 2000b]. A necessary condition to obtain such improvements is that the ensemble members be diverse. In additions, the individual predictors should be complementary, in the sense that each of them tends to make errors on different test instances.

Homogeneous ensembles are composed of classifiers of the same type. Ensembles composed of classifiers of different types are called *heterogeneous*. The strategies to generate diversity among the base classifiers are different for homogeneous and for heterogeneous ensembles. In homogeneous ensembles, the main difficulty is to generate diversity even when the same learning algorithm is used. To this end, one can use bootstrap techniques (e.g. bagging [Breiman, 1996c]), randomized steps in the base learning algorithm (e.g. the random subspace method used random forest [Breiman, 2001]), noise injection in the class labels (e.g. ECOC [Dietterich and Bakiri, 1991]) or adaptive emphasis protocols (e.g. boosting [Freund et al., 1999]). These techniques, which have exploited mainly in the context of homogeneous ensembles, can also be used to achieve further diversity in heterogeneous ensembles [Lu et al., 2015]. However, since different learning algorithms are used to generate the base learners, heterogeneous ensembles are intrinsically diverse. In this case, the main difficulty resides in determining the optimal way to combine the predictions of the different models in the ensemble.

Broadly speaking, the methods to build heterogeneous ensemble can be grouped into two categories. In the first family of methods a fixed number of different models are combined. A second strategy is to build a collection of models with different parametrizations and then select the best subset to include in the final ensemble. In [de Oliveira et al., 2013] a static heterogeneous ensemble is proposed. In this study 5 different base classifiers are combined: a Support Vector Machine (SVM), a multilayer perceptron (MLP), logistic regression, K nearest neighbors and decision tree. The parameters and

architecture of the individual classifiers are determined using 10-fold cross-validation. The proposed approach shows good results in the specific application of lithofacies classification. In [Nanni et al., 2015], a combination of several carefully optimized strong learners, such as deep neural networks, SVM, adaboosts, and gaussian processes, is proposed. The study shows a good performance of the proposed combination over several image classification and UCI tasks with respect to any of its constituents. However, the problem of determining of the number of classifiers of each type that need to be used is not solved in a fully satisfactory manner. Furthermore, the optimal composition of the ensemble is problem-dependent. A possible way to overcome this difficulty is to create a library of classifiers and then select a subset for the final ensemble [Caruana et al., 2004; Haque et al., 2016; Partalas et al., 2010]. For instance in [Caruana et al., 2004] a library of 2000 different methods trained with wide range of different parametrizations is build. The models included in the library are both individual classifiers and ensembles. The ensemble methods used include boosted trees using different decision tree algorithms and ensemble size, and bagged trees using different base decision trees. In addition, the individual trees of the bagged ensembles were also added to the library. Other individual classifiers included are SVMs trained with different parameters, multilayer perceptrons, etc. From that library of models, an iterative greedy selection algorithm is applied to build the final ensemble. The procedure starts with empty ensemble. Then, at each iteration the model that maximizes a performance measure (such as AUC or accuracy on a validation set) is included into the ensemble until all models in the library have been aggregated. Finally, the ensemble with the best performance in the validation set is selected as the final combination. Tsoumakas et al. have made several interesting contribution in this line of research [Partalas et al., 2010; Tsoumakas et al., 2004]. For instance, in [Partalas et al., 2010] the authors propose a greedy selection method from a library composed of 200 classifiers: 60 neural networks, 60 nearest neighbor classifiers, 80 SVMs and 20 decision trees. For each type of classifier, a parameter grid was defined and a single model was trained for each node in the grid. In their proposal, the ensemble is grown incrementally by selecting from the library one classifier at a time. At each step, the selection is made in terms of both individual accuracy and complementarity with the rest of the classifiers in the ensemble. In the problems investigated, such heterogeneous ensembles were found to be more accurate than their constituents. In [Haque et al., 2016] a genetic algorithm has been proposed to select the optimum structure of a heterogeneous ensemble from 20 different base models. These selection techniques, also known as ensemble pruning, have been also extensively applied to homogeneous ensembles [Tsoumakas et al., 2009].

In this work we propose to analyze heterogeneous ensembles in which the individual classifiers are selected from homogeneous ensembles. The goal is to build a family of

heterogeneous ensembles that can be smoothly transformed into each other. To this end, a family of heterogeneous ensembles of size T are built by pooling different fractions of base classifiers from M homogeneous ensembles of different types. Depending on the proportion of classifiers of each type, a particular heterogeneous combination in created. This family of heterogeneous ensembles can be represented in a regular simplex in M dimensions. The M vertices of this simplex represent the different homogeneous ensembles. The optimal fraction of each type of classifiers for the final ensemble is found by performing a search in this simplex.

The paper is organized as follows: In section 7.3, the design process to build optimal heterogeneous ensembles by pooling from homogeneous ensembles is described; Section 7.4, presents a comprehensive empirical evaluation of the proposed methodology and a comparison with the corresponding homogeneous ensembles and to individual classifiers. Finally, the conclusions of the present work are summarized.

7.3 From homogeneous to heterogeneous ensembles

In this study we analyze heterogeneous ensembles by pooling individual classifiers from different homogeneous ensembles. For this, we first train M ensembles of size T composed of M different types of base classifiers. The heterogeneous ensemble of size T is created by pooling $(t_1, t_2, \ldots, t_M)$ classifiers from the M ensembles, where t_j is the number of base classifiers pooled from the j^{th} homogeneous ensemble and $\sum_{j=1}^{M} t_j = T$. The optimum percentage of each type of base classifier can be obtained by cross-validation or out-of-bag error in a grid search in the space given by $(t_1, t_2, \ldots, t_M)$. Note, however, that there are $\binom{T+M-1}{M-1}$ different heterogeneous ensembles that can be built in this manner and that this number can be rather large even for small values of M and T. For instance, for $M = 3$ and $T = 101$, 5253 different heterogeneous ensembles can be built. In order to reduce the search space, the ensembles can be evaluated using intervals of i base classifiers of each type. For instance for $M = 3$, the followings configurations of the generated ensembles could be tested: $(0, 0, T)$, $(0, i, T - i)$, $(i, 0, T - i)$, $(0, 2i, T - 2i)$, $(2i, 0, T-2i)$, etc. This reduces the search space to $\binom{T/i+M-1}{M-1}$ possible ensemble configurations. Finally, the ensemble composition with minimum validation error is determined as the optimal ensemble. In the case that more than one ensemble configuration has the same minimum validation error, the average ensemble compositions for all minima with the same validation error is selected as the optimal heterogeneous ensemble.

For this study, we have used three homogeneous ensembles: random forests (RF), ensembles of support vector machines (SVM, [Cortes and Vapnik, 1995b]) and of multilayer perceptrons. All base classifiers of these ensembles are created using random samples

from the training set to allow for a fast validation of the optimum heterogeneous ensemble by means of out-of-bag validation [Breiman, 1996b]. In order to generate ensembles of SVMs the following randomized procedure is used. First, B sets of partially optimized parameters for the SVMs, $\mathbf{\Theta}_b$ with $b = 1, \ldots, B$, are obtained. More details on how these sets of partially optimized parameters are obtained are given below. Then, the ensemble is built in B batches of T/B SVMs. Each batch uses a different set of parameters $\mathbf{\Theta}_b$ and each individual SVMs is trained on a different random bootstrap sample without replacement of size 50% (i.e. subbagging) from the original training set. In this way the variability among the SVMs can be increased. Using subbagging has the advantage with respect to using standard bootstrap samples that the base models can be trained faster. This speedup is approximately 4 times considering the near quadratic training times of SVMs. In addition, the performance of both sampling strategies, bootstrapping and subbagging, has been demonstrated to be equivalent [Friedman and Hall, 2007; Martínez-Muñoz and Suárez, 2010]. To obtain the B sets of partially optimized parameters, we first define a parameter grid. Next, a subbagging sample is generated. One SVMs is trained for each combination of parameters and validated on the left-out set. Finally, the set of parameter with lower error is kept for building the ensemble. This process is repeated B times to obtain the $\mathbf{\Theta}_b$ with $b = 1, \ldots, B$ sets of parameters. The same procedure is used to generate the ensembles of MLPs. The training time complexity of the ensemble depends on the size of the parameter grid, B, T, on the sampling rate and on the complexity of the base classifier. Notwithstanding, in spite of creating an ensemble of SVMs (or MLPs), this procedure can be faster to train than training a single SVM by grid search and cross-validation, which is the most common way of training an SVM [Ben-Hur and Weston, 2010; Hsu et al., 2003]. In the next section we will show the validity of this procedure to generate homogeneous ensembles of SVMs and MLPs, and also of the procedure to obtain heterogeneous ensembles from them.

7.4 Experimental Results

In this section we present the empirical analysis of heterogeneous ensembles as the combination of homogeneous base classifiers. Furthermore, we validate the procedure to obtain SVM (and MLP) ensembles by partial optimization of their training parameters. We carried out the analysis on 19 datasets from the UCI repository [Bache and Lichman, 2013]. In all tested datasets, except of the synthetic problems, the training and test sets were generated using random stratified sampling with sizes 2/3 and 1/3 of the original sets respectively. In the synthetic classification problems, which are *Ringnorm*, *Threenorm* and *Twonorm*, 300 examples are sampled at random for training and 2000

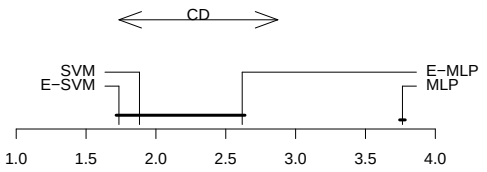


Fig. 7.1. Average ranks for SVM, E-SVM, MLP and E-MLP (more details in the text)

for testing using independent realizations. The results reported are averages over 100 executions except for *Breast Chess*, *German*, *Tic-tac-toe*, *Ozone* and *Spambase* were the averages are over 50 executions due to computational limitations.

Three, $M = 3$, homogeneous ensembles of size $T = 1001$ were trained. Specifically, the ensembles used are: standard random forest [Breiman, 2001], partially optimized ensemble of support vector machines and of multi layer perceptrons. We have used e1071, RSNNS and randomForest R packages for creating SVMs, MLPs and RF respectively. Under these setting the possible configurations of the heterogeneous ensemble are $1003 \times 1002/2$. To reduce the computational burden in the identification the optimum combination of base classifiers, we evaluated the heterogeneous ensembles in intervals of $i = 13$ base learners, which reduces the optimization to $78 \times 77/2$ evaluations. In addition, since the base classifiers of the three analyzed ensembles were generated using random subsamples from the training set, the optimum heterogeneous configuration is obtained by out-of-bag validation to further reduce the computational cost. The values of the hyperparameters for SVM with a RBF kernel are selected from a grid with $C = 2^q$ with $q = -5, \ldots, 15$ and $\boldsymbol{\gamma} = 2^p$ with $p = -15, \ldots, 3$. For MLP, the number of neurons in the hidden layer was optimized from the values $\{3, 4, 5, 6, 7, 8, 9, 10\}$. For building the partially optimized ensemble, $B = 10$ sets of hyperparameter were obtained using out-of-bag. For random forest, the default parameters were used.

7.4.1 Homogenous ensemble of SVMs and MLPs

In order to validate the procedure to generate the partially optimized ensembles, a comprehensive comparison with respect to an optimized single base learner was carried out. For this purpose, a single SVM and a single MLP were trained using within-train 10-fold cross-validation and grid search over the same sets of parameters given above. The average errors for this experiments are shown in Table 7.1 for a single SVM and MLP, and for the homogeneous ensembles composed of SVMs (shown as E-SVM in the

Table 7.1
Test errors for a single optimized SVM and MLP, also their homogeneous ensembles as it is proposed in section 7.4.1

Dataset	SVM	E-SVM	MLP	E-MLP
Australian	14.36±2.3	**13.69±2.1***	15.62±2.2	14.16±1.9
Boston	**12.18±2.4**	12.20±2.3	12.65±2.1	12.31±2.0
Breast	3.47±1.1	3.37±1.1	5.06±1.7	**3.24±1.1**
Bupa	29.05±3.7	**27.85±3.4**	30.26±4.0	28.32±3.7
Chess	**0.76±0.4**	0.81±0.3	0.98±0.2	0.92±0.3
Colic	31.79±3.4	33.20±1.4	32.37±3.4	**31.30±3.3**
German	25.05±1.8	**24.59±1.6**	28.02±2.0	24.70±1.9
Heart	15.99±3.5	**15.38±3.0***	18.19±3.7	16.34±3.1
Hepatitis	16.56±3.6	15.83±3.0	17.46±4.3	**15.42±4.3**
Ionosphere	6.26±1.8	**5.73±1.7***	10.56±2.9	11.28±2.5
Ozone	5.65±0.4	5.60±0.3	6.75±0.5	**5.48±0.5**
Parkinsons	**8.74±4.1***	10.71±3.7	11.29±4.1	13.66±3.9
Pima	23.05±2.0	**22.68±1.8***	24.73±2.4	23.10±2.1
Ringnorm	1.74±0.6	**1.58±0.4***	17.02±1.5	16.41±1.5
Spambase	6.46±0.4	6.63±0.4	7.05±0.4	**5.90±0.4***
Sonar	**14.96±4.3***	17.78±4.9	20.99±4.2	20.65±4.6
Threenorm	14.54±1.3	**14.10±0.7***	17.70±2.0	16.93±0.9
Tic-tac-toe	**1.00±1.3***	1.83±0.7	4.38±7.4	1.83±0.7
Twonorm	2.63±0.5	**2.44±0.3***	4.02±0.69	2.94±0.4

table) and of MLPs (shown as E-MLP). For each dataset, the best method is highlighted in boldface and the second best method is underlined. In addition, an overall comparison of the methods is shown in Figure 7.1 by mean of the procedure proposed by Demšar in [Demsar, 2006]. In this diagram, the average ranks for each method are shown. Methods connected by a horizontal solid line indicate that their differences in average rank are not statistically significant according to a Nemenyi test (p-value < 0.05).

From Table 7.1, it can be observed that the ensemble of MLPs clearly outperforms the single MLP. The differences are favourable to the ensemble of MLPs except for *Ionosphere* and *Parkinsons*. The differences between the single SVM and its ensemble counterpart are not so pronounced as the ones observed for MLPs. The ensemble of SVM obtains a better result than a single SVM in 12 out of 19 datasets. This same result can be observed in Figure 7.1 where the average rank of E-SVM is slightly better than a single SVM. However, the difference is not statistically significant. Even though the differences are not statistically significant, this analysis shows that this procedure to build ensembles of SVMs is not detrimental. When using MLP as the base classifiers,

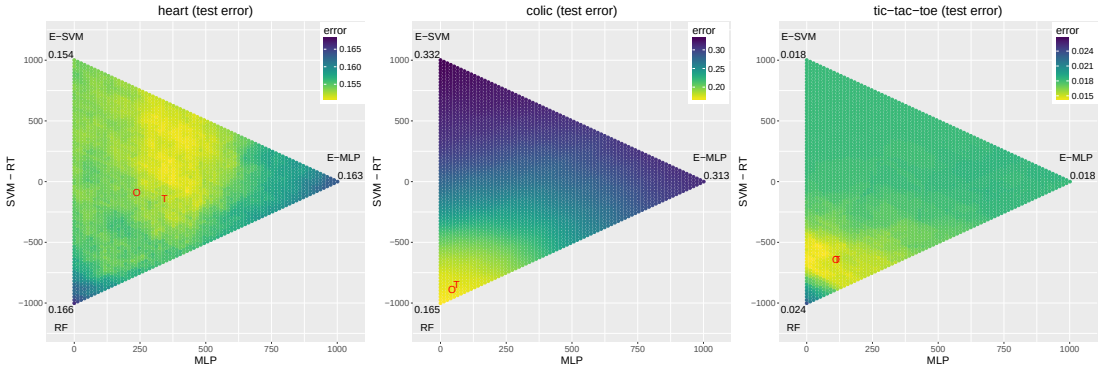


Fig. 7.2. Test error rate of the heterogenous ensembles in the simplex for different classification problems. Darker colors correspond to higher errors

Table 7.2
Test errors of single classifiers, homogeneous ensembles and optimal heterogeneous ensemble (I)

Dataset	E-SVM	E-MLP	RF	SIM	[% SVM, % MLP, % Trees]	entropy/max
Australian	13.69±2.1	14.16±1.9	**13.02±2.1***	13.51±2.0	[24.5 , 16.7 , 58.8]	0.87
Boston	**12.20±2.3**	12.31±2.0	12.85±2.1	12.23±2.0	[39.2 , 23.1 , 37.7]	0.98
Breast	3.37±1.1	**3.24±1.1**	3.30±1.1	3.30±1.0	[27.6 , 29.0 , 43.4]	0.98
Bupa	27.85±3.4	28.32±3.7	**27.17±3.6**	27.27±3.5	[21.4 , 15.6 , 63.0]	0.83
Chess	0.81±0.3	0.92±0.3	1.72±0.4	**0.76±0.2**	[34.7 , 22.7 , 42.6]	0.97
Colic	33.20±1.4	31.30±3.3	**16.53±2.9***	17.20±3.0	[3.7 , 4.4 , 91.9]	0.3
German	24.59±1.6	24.70±1.9	**23.94±1.8***	24.31±1.9	[16.0 , 28.1 , 55.8]	0.89
Heart	**15.38±3.0**	16.34±3.1	16.62±2.9	15.50±3.1	[33.5 , 23.8 , 42.7]	0.97
Hepatitis	15.83±3.0	15.42±4.3	**15.12±3.6**	15.19±3.6	[25.6 , 28.7 , 45.7]	0.97
Ionosphere	**5.73±1.7**	11.28±2.5	6.69±1.7	5.84±1.7	[64.4 , 13.7 , 21.9]	0.81
Ozone	5.60±0.3	5.48±0.5	5.67±0.3	**5.37±0.4**	[17.8 , 52.8 , 29.4]	0.91
Parkinsons	**10.71±3.7**	13.66±3.9	11.08±4.0	**10.71±3.9**	[44.1 , 12.9 , 43.0]	0.9
Pima	**22.68±1.8***	23.10±2.1	23.12±2.0	22.95±1.8	[44.5 , 18.1 , 37.4]	0.94
Ringnorm	**1.58±0.4***	16.41±1.5	5.87±1.0	1.70±0.5	[62.2 , 11.2 , 26.6]	0.81
Spambase	6.63±0.4	5.90±0.4	5.11±0.4	**4.97±0.3**	[12.2 , 11.1 , 76.8]	0.64
Sonar	**17.78±4.9**	20.65±4.6	18.88±4.8	17.97±4.4	[39.1 , 16.9 , 44.0]	0.94
Threenorm	**14.10±0.7***	16.93±0.9	16.67±1.0	14.36±0.8	[52.1 , 10.8 , 37.1]	0.86
Tic-tac-toe	1.83±0.7	1.83±0.7	2.35±1.1	**1.46±0.7***	[12.5 , 11.2 , 76.3]	0.65
Twonorm	**2.44±0.3***	2.94±0.4	3.90±0.5	2.55±0.4	[42.5 , 22.7 , 34.8]	0.97

we observe that the differences are statistically significant with respect to a single MLP. In addition, with these settings, we have observed that the training time for E-SVM is about 2 times faster than training a single SVM using grid search and 10-fold cross-validation. On the other hand, the ensemble of MLPs is about 10 times slower than the single MLP due to the linear complexity of MLP with respect to the number of training instances.

7.4.2 Heterogeneous ensemble pooled from homogeneous ensembles

In this section the performance of the proposed procedure to built heterogeneous ensembles by pooling from homogeneous ensembles is analyzed. The objective is to find

the optimum proportion of each of the possible base classifiers to include in the final heterogeneous ensemble. Each of the possible selected proportions, which correspond to a different heterogeneous ensemble, can be represented by a point in a regular simplex in M dimensions. This is shown in Figure 7.2 for three representative datasets: *Heart*, *Colic* and *Tic-tac-toe*. Each plot in Figure 7.2 shows in a 3 dimensional simplex, the average test error for the different combinations of base classifiers in intervals of $i = 13$ classifiers using a color map scale scheme. Darker colors indicate higher average error as indicated by the color legend at the top-right of each plot. The three vertices in the plots correspond to the three tested homogeneous ensembles. The vertices in the upper left, right and bottom left of the plot correspond to E-SVM, E-MLP and random forest respectively. A displacement away from one of these vertices smoothly transforms the corresponding homogeneous ensembles into a heterogeneous one. The horizontal axis shows the number of selected MLPs in the heterogeneous ensemble, while the vertical axis indicates the number of SVMs minus the number of random trees. In addition, all plots show the average of the selected positions using out-of-bag validation (marked with a 'o' sign) and the average position for the best test errors (marked with a 'T' sign).

In the plots of Figure 7.2 different behaviours of the combination of base classifiers can be observed. In *Heart* (left plot), the best position is observed quite centered, showing that a heterogeneous ensemble composed of base classifiers from different types is beneficial to improve the generalization performance of the ensemble. However, this is not a general trend as it can be observed in the center plot (*Colic*). In this case, the best result is clearly located at one of the vertices of the simplex that correspond to a homogeneous ensemble of random forest in this case. Finally, it is important to note that the optimum location need not be close to the best homogeneous ensemble. For instance, in *Tic-tac-toe*, the location of the minimum error is very close to the random forest vertex in spite of the fact that this homogeneous ensemble presents the worst average performance. Finally, we can observe that the average location of the minima identified using out-of-bag is quite close to the location in test. We have also observed, however, that for the smaller datasets the identification of the optimum point is less accurate.

In the Table 7.2, the average test errors for the homogeneous ensembles of SVMs (E-SVM) and MLPs (E-MLP), random forest (RF) and the proposed strategy (SIM) over the investigated problems are reported. The best and second best results for each dataset are highlighted in boldface and underlined respectively. In addition, the table shows the average percentage of classifiers of each type selected by out-of-bag validation for the heterogeneous ensembles. The percentages are shown in the same order that the ensembles are shown, that is, % of SVMs, % of MLP and % of random trees. The last

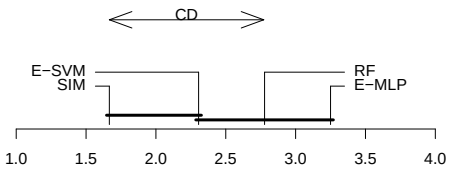


Fig. 7.3. Average ranks for E-SVM, E-MLP, RF and the optimal estimated heterogeneous ensemble

column of the table indicates the entropy of the selected percentages of classifiers divided by the maximum entropy (i.e. [33.3, 33.3, 33.3]).

As shown in Table 7.2, the proposed method is the best or the second best method for all datasets. E-SVM also achieves rather good results but it is somehow less consistent. E-SVM is the method that obtains the highest number of best performances (in 9 datasets) but its performance is the worst in 4 datasets. Finally, random forest and E-MLP obtain 5 and 1 best results respectively. This results are summarized using a Demšar plot [Demsar, 2006] in Figure 7.3. From this diagram, it can be observed that the proposed procedure is significantly better than random forest and E-MLP (as given by a Nemenyi test with p-value < 0.05). The proposed methodology has an average rank better that E-SVM but the difference is not statistically significant.

7.5 Conclusions

In this study, a continuous family of heterogeneous ensembles of size T with varying proportions of base classifiers of different types is analyzed. To this end, we first generate M different homogeneous ensembles. Diversification in these ensembles is obtained by using both subsampling and randomization techniques. Then a heterogenous ensemble is built by pooling classifiers from these homogeneous ensembles. The proportions of classifiers of different types in the heterogeneous combination can be represented with a point in a simplex in M dimensions. Each of the M vertices in this simples corresponds to one of the homogeneous ensembles. The optimal proportion of base classifiers in the final ensemble, which is strongly problem-dependent, can be estimated using out-of-bag data.

In the empirical evaluation carried out, the proposed strategy consistently exhibits excellent performance. In the problems investigated, it is either the first or second most accurate method. The results show that the proposed combination is better that any of

the homogeneous ensembles; i.e. random forest, ensembles of MLPs and ensembles of SVMs. In addition, the differences of average ranks are statistically significant except for the ensemble of SVMs, which is second best.

Chapter 8

Conclusions and Future Work

As a result of the research carried out in this thesis a series of ensemble learning methods have been designed, analyzed and evaluated. The proposed methods result in improvements of the generalization accuracy in noisy learning tasks and, in some cases, a more effective use of computational resources. The proposed ensemble techniques can be divided into three categories: robust ensembles in noisy domains, noise detection methods, and ensembles of strong learners.

Specifically, in the first contribution of this thesis, we have used bootstrap-based ensemble methods, such as bagging or random forest, to design a preprocessing strategy for noise detection. In these methods, subsampling is used as a regularization technique. In bootstrap ensembles the optimal sampling rate can be determined using out-of-bag data. Employing a wrapper method, the examples that are misclassified by more than a certain fraction of the base learners are marked as noise. The optimal threshold depends on the type of problem and the level of noise that is present in the dataset. Therefore, cross-validation has been used to determine the optimal threshold for each problem. Finally, the detected noisy examples can be either removed (filtered) or its class corrected (cleaned). Filtering is found to be slightly superior in low or medium noise levels. However, when the noise level is high, the removal of examples that are likely to be noisy causes a significant loss of information. Hence, for high noise levels, cleaning seems to be preferable. The proposed noise detection strategy has been evaluated in 19 binary classification problems with different levels of injected noise (0%, 10%, 20%, 30% and 40%). The proposed method is an effective noise detection strategy specially in the presence of high levels of class-label noise. In particular, it outperforms the commonly used majority or consensus filtering strategies.

The second contribution is the design of ensembles that are robust to class-label noise. Subsampling is also used as an effective regularization mechanism. Using small sampling

rates in bootstrap ensembles generally increases the diversity among the base learners. Typically, the negative effect of noisy examples is reduced when small bootstrap samples are used to build the individual ensemble classifiers. An interesting observation is that subsampling increases the robustness of bagging in spite of decreasing the classification margins. For random forest, oversampling can in fact be more effective than subsampling. This is probably a consequence of the inherent diversity present in this type of randomized ensemble.

One of the main contributions of this thesis is vote-boosting, a novel boosting algorithm based on determining the weights of the training instances in the learning process in terms the degree of uncertainty of the ensemble prediction. Since the class labels are not taken into account in the emphasis strategy, this algorithm does not suffer from the sensitivity of standard boosting algorithms (e.g. AdaBoost) to class-label noise. The emphasis function used in this study is the symmetric beta distribution. Depending on the value of the shape parameters of this distribution the learning progressively focuses on instances on which the ensemble classifiers agree or disagree. In problems where noise level is low and when simple base learners are used, it is more effective to emphasize instances for which the disagreement rate is high. For flexible classifiers, the emphasis on these uncertain instances should be reduced. In problems with high levels of class-label noise, the focus should be on instances on which the ensemble classifiers agree. The optimal shape parameters for the emphasis function can be determined using cross-validation. Vote-boosting can be viewed as a gradient descent optimization in the functional space of linear combinations of base learners. The proposed method has two important advantages with respect to AdaBoost. Vote-boosting can generate ensembles that are both accurate and robust to class labels noise. Furthermore, it can be used to improve the performance of base learners whose training error is small (e.g. random trees)

A requirement for building effective ensembles is the use of classifiers whose predictions are complementary. However, when strong classifiers such as SVMs are used, because of the stability of their predictions, it is difficult to achieve a sufficient level of diversity. In this thesis we propose to use a combination randomization and optimization techniques to increase the diversity of SVM ensembles. Specifically, three different approaches to building ensembles of SVMs are considered: The completely-optimized SVM ensemble (COSE), the partially-optimized SVM ensemble (POSE), and the randomized-optimized SVM ensemble (ROSE). In COSE, the SVM hyperparameters are optimized for each individual. In POSE, we optimize the SVM hyperparameters for B (10 in this study) bootstrap samples of the training data. In ROSE, B hyperparameters are chosen by selecting the best among a number of randomly generated sets. In both POSE and ROSE the selected hyperparameters are reused to build different SVMs on bootstrap samples of

the original training data. The accuracies of these ensembles are comparable to a single SVM: COSE is slightly more accurate than a single SVM; POSE's accuracy is similar to a single SVM and higher than ROSE. COSE ensembles are costly to generate. However, building POSE and ROSE ensembles is between 2 and 10 times faster than training a single SVM whose hyperparameters are determined using standard grid search with 10-folds cross validation. In principle, these techniques can be used to build ensembles of other types of strong classifiers, such as deep neural networks.

Finally, we proposed a strategy for building heterogeneous ensembles that perform a sort of interpolation among homogeneous ensembles of different types. Different types of base learners offers the advantage of having diverse models with different and, hopefully, complementary biases. The starting point is a collection of homogeneous ensembles of feedforward multilayer perceptrons (MLPs), support vector machines (SVMs), and random trees (RTs). Different heterogeneous combinations of classifiers from these ensembles can be represented as points in a three-dimensional regular simplex. Each of the vertices of this simplex corresponds to one of the homogeneous ensembles. The optimal proportion of base classifiers in the final ensemble can be estimated using out-of-bag data. The proposed method shows better accuracy in relation to its homogeneous counterparts.

8.1 Future Work

One of the main contributions of this thesis is a novel boosting method, vote-boosting, in which the level agreement or disagreement among the predictions of the ensemble members is used to determine the evolution of the instance weights. The emphasis function considered is the symmetric beta distribution. In binary classification problems with class-imbalance, the scarcity of examples in the minority class can result in a bias toward predicting more frequently the majority class. In such cases it could be of some advantage to consider asymmetric probability densities for emphasis. The use of an asymmetric emphasis would tend to give higher weights to the examples in the minority class. This biased emphasis can be useful to compensate the shortage of the examples in that class. An additional improvement for vote-boosting, which is interesting to explore, is to adapt the emphasis function to multi-class classification problems. This can be done in a quite straightforward manner using a multivariate probability distribution such as the Dirichlet distribution.

Recently, Gradient boosting and in particular its XGBoost implementation, has gained attention in different applications. However, it has a high tendency to overfitting. Specifically, the iterative fitting of the pseudo-residuals can be misleading for this sequential

ensemble method. One of the interesting ideas to investigate, is to implement vote-boosting within the gradient boosting framework. The pseudo-residuals should be replaced for a measure that can be calculated in terms of agreement (or disagreement) among base learners, without considering class labels. The logistic (binary classification) or the softmax (multiclass classification) transformation can be applied to transform the output of the regression trees into estimates of class probabilities.

Another area of interest is the determination of the SVM hyperparameters in a more efficient way. A faster search strategy to select good values of the hyperparameters space would result in a large reduction of the training costs. Our experiments show that, even though performance of the SVM is very sensitive to the values of the hyperparameters, there are sets of values of the hyperparameters all of which work reasonably well. One of the ideas is to identify regions (of various sizes) within which choosing random pairs of hyperparameters work equally well. First, several disperse points are selected in the search space. Then, for each pair, a few random validation test can be done in its neighborhood. The points with lower validation accuracy and its neighborhood can be discarded from the search space. This random search can be used to identify pairs of hyperparameters that are good candidates. The values of the hyperparameters can then be tuned using a local search within the remained pairs in the search space.

Finally, one of the contributions of this thesis is a method to build an effective heterogeneous ensemble by pooling base learners from different homogeneous ensembles. In the presented method, we searched for an optimal heterogeneous ensemble in a regular simplex in M dimensions, where M (3 in the proposed method) is the number of homogeneous ensembles. Using base learners of different types increases the chance of having complementarity predictions. Extending the presented approach so that it includes more base learners from different types and, specially, designing an efficient search strategy in the simplex are interesting topics to investigate.

Chapter 9

Conclusiones y trabajo futuro

Como resultado de la investigación llevada a cabo en esta tesis se han diseñado, analizado y evaluado, una serie de métodos de conjuntos de clasificadores. Los métodos propuestos resultan en mejoras en la precisión de la generalización en tareas de aprendizaje ruidosas y, en algunos casos, un uso más efectivo de los recursos computacionales. Las técnicas de conjunto propuestas se pueden dividir en tres categorías: Conjuntos robustos en dominios ruidosos, métodos de detección de ruido y conjuntos de clasificadores fuertes.

En concreto, en la primera contribución de esta tesis, hemos utilizado métodos de conjunto basados en bootstrap, como bagging o random forest, para diseñar una estrategia de preprocesamiento de detección de ruido. En estos métodos, el submuestreo se utiliza como una técnica de regularización. En los conjuntos de bootstrap, la tasa de muestreo óptima se puede determinar utilizando datos *out-of-bag*. Al emplear un método *wrapper*, los ejemplos que están clasificados erróneamente por más de una fracción dada de clasificadores base se marcan como ruido. El umbral óptimo depende del problema y del nivel de ruido presente en el conjunto de datos. Validación cruzada se ha utilizado para determinar el umbral óptimo para cada problema. Finalmente, los ejemplos ruidosos detectados pueden eliminarse (filtrarse) o corregir (limpiar) su clase. Filtrar los ejemplos es ligeramente superior a limpiarlos en niveles de ruido bajos o medios. Sin embargo, cuando el nivel de ruido es alto, la eliminación de ejemplos que probablemente sean ruidosos provoca una pérdida de información. Por lo tanto, para niveles elevados de ruido, la limpieza parece ser preferible. La estrategia de detección de ruido propuesta se ha evaluado en 19 problemas de clasificación binaria con diferentes niveles de ruido inyectado (0 %, 10 %, 20 %, 30 % y 40 %). El método propuesto es una estrategia efectiva de detección de ruido, especialmente en presencia de altos niveles de ruido de clase. En particular, supera a las estrategias de filtrado por consenso o por mayoría comúnmente utilizados.

La segunda contribución de esta tesis es el diseño de conjuntos robustos a ruido en etiquetas de clase. El submuestreo también se utiliza como un mecanismo de regularización eficaz. El uso de submuestreo reducido en conjuntos de bootstrap generalmente aumenta la diversidad entre los clasificadores de base. Por lo general, el efecto negativo de los ejemplos ruidosos se reduce cuando se utilizan pequeñas muestras para construir los clasificadores individuales del conjunto. Una observación interesante es que el submuestreo aumenta la robustez de bagging a pesar de disminuir los márgenes de clasificación. Para random forests, el sobremuestreo puede de hecho ser más efectivo que el submuestreo. Esto es probablemente una consecuencia de la diversidad inherente presente en este tipo de conjunto.

Una de las principales contribuciones de esta tesis es vote-boosting, un nuevo algoritmo de boosting basado en la determinación de los pesos de las instancias de entrenamiento en el proceso de aprendizaje. En términos del grado de incertidumbre de la predicción de conjunto. Dado que las etiquetas de clase no se tienen en cuenta en esta estrategia de énfasis, este algoritmo no sufre de la sensibilidad de los algoritmos de boosting estándar (por ejemplo, AdaBoost) al ruido de clase. La función de énfasis utilizada en este estudio es la distribución beta simétrica. Dependiendo del valor de los parámetros de la distribución, el aprendizaje se centra progresivamente en los casos en los que los clasificadores del conjunto están de acuerdo o en desacuerdo. En los problemas en los que el nivel de ruido es bajo y cuando los classificadores base son sencillos, es más efectivo enfatizar los casos para los cuales la tasa de desacuerdo es alta. Para clasificadores base más complejos, el énfasis en las instancias inciertas debe reducirse. En los problemas con altos niveles de ruido de etiqueta de clase, el énfasis debe estar en los casos en que los clasificadores de conjunto estén de acuerdo. Los parámetros de forma óptimos para la función de énfasis se pueden determinar mediante la validación cruzada. El aumento de votos puede verse como una optimización de pendiente de gradiente En el espacio funcional de combinaciones lineales de aprendices base. El método propuesto tiene dos ventajas importantes con respecto a AdaBoost. El aumento de votos puede generar conjuntos que sean precisos y robustos. Además, se puede utilizar para mejorar el rendimiento de los clasificadores base con bajo error en entrenamiento (por ejemplo, árboles aleatorios).

Un requisito para construir conjuntos efectivos es el uso de clasificadores con predicciones complementarias. Sin embargo, cuando se usan clasificadores *fuertes* como SVMs es difícil lograr un nivel de diversidad suficiente. En esta tesis, proponemos utilizar una combinación de técnicas de aleatorización y optimización para aumentar la diversidad de los conjuntos de SVM. En concreto, se consideran tres enfoques diferentes para construir conjuntos de SVM: El conjunto de SVMs completamente optimizado (COSE), el conjunto de SVMs parcialmente optimizado (POSE) y el conjunto de SVMs optimizado al

azar (ROSE). En COSE, los hiperparámetros SVM se obtienen para cada individuo. En POSE, optimizamos los hiperparámetros de la SVM para B (10 en este estudio) muestras bootstrap. En ROSE, los hiperparámetros B se eligen seleccionando los mejores hiperparámetros de entre una serie generados aleatoriamente. Tanto en POSE como en ROSE, los hiperparámetros seleccionados se reutilizan para crear diferentes SVM en las muestras bootstrap. El error de generalización de estos conjuntos es comparable a una sola SVM. En concreto para COSE es la precisión es ligeramente más mayor que para una sola SVM; En POSE es similar y en ROSE es superior. Sin embargo, los conjuntos COSE son costosos de generar, mientras que la creación de conjuntos POSE y ROSE es entre 2 y 10 veces más rápida que entrenar una SVM donde los hiperparámetros se determinan mediante búsqueda en cuadrícula con una validación cruzada de 10 pliegues. En principio, estas técnicas se podrían utilizar para crear conjuntos de otros tipos de clasificadores fuertes, como las redes neuronales profundas.

Finalmente, propusimos una estrategia para construir conjuntos heterogéneos como una interpolación entre conjuntos homogéneos de diferentes tipos. Los diferentes tipos de aprendices básicos ofrecen la ventaja de tener diversos modelos con sesgos diferentes y, posiblememnte, complementarios. El punto de partida es una colección de conjuntos homogéneos de perceptrones multicapa (MLP), máquinas de vectores soporte (SVM) y árboles aleatorios (RT). Los conjuntos se pueden representar como puntos en un simplex. Cada uno de los vértices de este simplex corresponde a uno de los conjuntos homogéneos. La proporción óptima de clasificadores base en el conjunto final se puede estimar utilizando datos *out-of-bag*. El método propuesto muestra una mayor precisión en relación con sus homólogos homogéneos.

9.1 Trabajo futuro

Una de las principales contribuciones de esta tesis es un novedoso método de booting, vote-boosting, en el que el acuerdo o desacuerdo entre las predicciones de los miembros del conjunto se utiliza para determinar la evolución de los pesos de las instancias. La función de énfasis considerada es la distribución beta simétrica. En los problemas de clasificación binaria con desequilibrio de clases, la escasez de ejemplos en la clase minoritaria puede dar lugar a un sesgo hacia la predicción de la clase mayoritaria con mayor frecuencia. En tales casos, podría ser una ventaja considerar densidades de probabilidad asimétricas para el énfasis de los datos. El uso de un énfasis asimétrico tendería a dar mayor peso a los ejemplos en la clase minoritaria, lo que puede ser útil para compensar la escasez de ejemplos en esa clase. Una mejora adicional para vote-booting, que podría ser interesante de explorar, es adaptar la función de énfasis a problemas de clasificación

de múltiples clases. Esto se puede hacer de una manera bastante sencilla usando la distribución Dirichlet de probabilidad.

En investigaciones reciente, gradient boosting y en particular su implementación XGBoost, ha ganado atención en diferentes aplicaciones. Sin embargo, tiene una alta tendencia al sobreajuste. En particular, el ajuste iterativo de los pseudo-residuos puede caer en sobreajuste. Una de las ideas interesantes para investigar, es implementar el vote-boosting dentro del marco gradient boosting. Los pseudo-residuos deberían reemplazarse por una medida que se pueda calcular en términos de acuerdo (o desacuerdo) entre los clasificadores base, sin tener en cuenta las etiquetas de clase. Se puede aplicar la transformación logística (clasificación binaria) o softmax (clasificación multiclase) para transformar la salida de los árboles de regresión en estimaciones de probabilidades de clase.

Otra área de interés es la determinación de los hiperparámetros de una SVM de manera más eficiente. Una estrategia de búsqueda más rápida para seleccionar buenos valores del espacio de hiperparámetros daría lugar a una gran reducción de los costes de entrenamiento. Nuestros experimentos muestran que, aunque el rendimiento de la SVM es muy sensible a los valores de los hiperparámetros, hay combinaciones de valores de los hiperparámetros que funcionan razonablemente bien. Una de las ideas sería identificar regiones (de varios tamaños) dentro de las cuales la elección de pares aleatorios de hiperparámetros funcione de forma similar. Primero, se seleccionan varios puntos dispersos en el espacio de búsqueda. Luego, para cada par, se puede hacer una prueba de validación aleatoria en su vecindad. Los puntos con menor precisión y su vecindario se pueden descartar del espacio de búsqueda. Esta búsqueda aleatoria se puede utilizar para identificar pares de hiperparámetros que son buenos candidatosi. Posteriormente los valores de los hiperparámetros se pueden ajustar usando una búsqueda local dentro de los pares restantes en el espacio de búsqueda.

Finalmente, una de las contribuciones de esta tesis es un método para construir un conjunto heterogéneo efectivo mediante la agrupación de aprendices base de diferentes conjuntos homogéneos. En el método presentado, se buscó un conjunto heterogéneo óptimo en un símplex regular en dimensiones M, donde M (3 en el método propuesto) es el número de conjuntos homogéneos. El uso de aprendices base de diferentes tipos aumenta la posibilidad de tener predicciones complementarias. Ampliar el enfoque presentado para que incluya aprendices base de diferentes tipos y, especialmente, el diseño de una estrategia de búsqueda eficiente en el simplex son temas interesantes para investigar.

Bibliography

Naoki Abe, Bianca Zadrozny, and John Langford. Outlier detection by active learning. In *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 767–772, New York, NY, USA, 2006. ACM Press.

Joaquín Abellán and Andrés R Masegosa. Bagging decision trees on data sets with classification noise. In *Foundations of Information and Knowledge Systems*, pages 248–265. Springer, 2010.

Muhammad Zeshan Afzal, Samuele Capobianco, Muhammad Imran Malik, Simone Marinai, Thomas M Breuel, Andreas Dengel, and Marcus Liwicki. Deepdocclassifier: Document classification with deep convolutional neural network. In *Document Analysis and Recognition (ICDAR), 2015 13th International Conference on*, pages 1111–1115. IEEE, 2015.

Anas Ahachad, Adil Omari, and Aníbal R. Figueiras-Vidal. Neighborhood guided smoothed emphasis for real adaboost ensembles. *Neural Processing Letters*, 42(1): 155–165, 2015.

Anas Ahachad, Lorena Álvarez Pérez, and Aníbal R. Figueiras-Vidal. Boosting ensembles with controlled emphasis intensity. *Pattern Recognition Letters*, 88:1 – 5, 2017.

Esteban Alfaro, Matías Gámez, and Noelia García. adabag: An R package for classification with boosting and bagging. *Journal of Statistical Software*, 54(2):1–35, 2013.

Kamal Ali. On the link between error correlation and error reduction in decision tree ensembles. Technical report, 1995.

Kamal M. Ali and Michael J. Pazzani. Error reduction through learning multiple descriptions. *Machine Learning*, 24(3):173–202, Sep 1994.

Naomi S Altman. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46(3):175–185, 1992.

Giuliano Armano and Emanuele Tamponi. Building forests of local trees. *Pattern Recognition*, 76:380 – 390, 2018.

K. Bache and M. Lichman. UCI machine learning repository, 2013.

Robert E. Banfield, Lawrence O. Hall, Kevin W. Bowyer, and W. Philip Kegelmeyer. Ensemble diversity measures and their application to thinning. *Information Fusion*, 6:2005, 2004.

Robert E. Banfield, Lawrence O. Hall, Kevin W. Bowyer, and W. Philip Kegelmeyer. A comparison of decision tree ensemble creation techniques. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(1):173–180, 2007.

Peter Bartlett and John Shawe-taylor. Generalization performance of support vector machines and other pattern classifiers, 1998.

Eric Bauer and Ron Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning*, 36(1):105–139, Jul 1999.

Asa Ben-Hur and Jason Weston. *A user's guide to support vector machines*, pages 223–239. Humana Press, Totowa, NJ, 2010.

Simon Bernard, Laurent Heutte, and Sébastien Adam. Influence of hyperparameters on random forest accuracy. In *Multiple Classifier Systems, 8th International Workshop, MCS 2009, Reykjavik, Iceland, June 10-12, 2009. Proceedings*, pages 171–180, 2009.

Marcelo Bertalmio, Luminita Vese, Guillermo Sapiro, and Stanley Osher. Simultaneous structure and texture image inpainting. *IEEE transactions on image processing*, 12 (8):882–889, 2003.

Jinbo Bi and Tong Zhang. Support vector classification with input data uncertainty. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 161–168. MIT Press, 2005.

Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, COLT '92, pages 144–152, New York, NY, USA, 1992. ACM. ISBN 0-89791-497-X. doi: 10.1145/130385.130401. URL http://doi.acm.org/10.1145/130385.130401.

Léon Bottou and Chih-Jen Lin. Support vector machine solvers. *Large scale kernel machines*, pages 301–320, 2007.

Remco R. Bouckaert and Eibe Frank. *Evaluating the replicability of significance tests for comparing learning algorithms*, pages 3–12. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004. ISBN 978-3-540-24775-3.

L. Breiman. Arcing classifiers. *Annals of Statistics*, 26:801–823, 1998.

Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996a.

Leo Breiman. Out-of-bag estimation. Technical report, Statistics Department, University of California, 1996b.

Leo Breiman. Stacked regressions. *Machine learning*, 24(1):49–64, 1996c.

Leo Breiman. Bias, variance, and arcing classifiers. Technical Report 460, Statistics Department, University of California, 1996d.

Leo Breiman. Prediction games and arcing algorithms. *Neural computation*, 11(7): 1493–1517, 1999.

Leo Breiman. Randomizing outputs to increase prediction accuracy. *Machine Learning*, 40(3):229–242, 2000.

Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, Oct 2001.

Leo Breiman, Jerome Friedman, Charles J. Stone, and R.A. Olshen. *Classification and Regression Trees*. The Wadsworth and Brooks-Cole statistics-probability series. Taylor & Francis, 1984. ISBN 9780412048418.

Carla E. Brodley and Mark A. Friedl. Identifying mislabeled training data. *Journal of Artificial Intelligence Research*, 11(1):131–167, July 1999. ISSN 1076-9757.

Gavin Brown, Jeremy Wyatt, Rachel Harris, and Xin Yao. Diversity creation methods: a survey and categorisation. *Information Fusion*, 6(1):5 – 20, 2005. Diversity in Multiple Classifier Systems.

Peter Bühlmann and Bin Yu. Analyzing bagging. *Annals of Statistics*, 30(4):927–961, 08 2002.

Andreas Buja and Werner Stuetzle. Observations on bagging. *Statistica Sinica*, 16(2): 323, 2006.

Christopher J.C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

Jingjing Cao, Sam Kwong, and Ran Wang. A noise-detection based adaboost algorithm for mislabeled data. *Pattern Recognition*, 45(12):4451 – 4465, 2012.

Rich Caruana and Alexandru Niculescu-Mizil. An empirical comparison of supervised learning algorithms. In *Machine Learning, Proceedings of the Twenty-Third International Conference (ICML 2006), Pittsburgh, Pennsylvania, USA, June 25-29, 2006*, pages 161–168, 2006.

Rich Caruana, Alexandru Niculescu-Mizil, Geoff Crew, and Alex Ksikes. Ensemble selection from libraries of models. In *Proceedings of the Twenty-first International Conference on Machine Learning*, ICML '04, pages 18–, New York, NY, USA, 2004. ACM. ISBN 1-58113-838-5.

Rich Caruana, Nikolaos Karampatziakis, and Ainur Yessenalina. An empirical evaluation of supervised learning in high dimensions. In *Machine Learning, Proceedings of the Twenty-Fifth International Conference (ICML 2008), Helsinki, Finland, June 5-9, 2008*, pages 96–103, 2008.

Sunsern Cheamanunkul, Evan Ettinger, and Yoav Freund. Non-convex boosting overcomes random label noise. *Computing Research Repository*, abs/1409.2905, 2014.

Shyi-Ming Chen and Jeng-Ren Hwang. Temperature prediction using fuzzy time series. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 30(2): 263–275, 2000.

Tianqi Chen, Tong He, Michael Benesty, et al. Xgboost: extreme gradient boosting. *R package version 0.4-2*, pages 1–4, 2015.

V. Cherkassky and Y. Ma. Practical selection of svm parameters and noise estimation for svm regression. *Neural Network*, 17(1):113–126, 2004. ISSN 0893-6080.

Marc Claesen, Frank De Smet, Johan A.K. Suykens, and Bart De Moor. Ensemblesvm: A library for ensemble learning using support vector machines. *Journal of Machine Learning Research*, 15:141–145, 2014. URL http://jmlr.org/papers/v15/claesen14a.html.

Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20 (3):273–297, Sep 1995a. ISSN 1573-0565. doi: 10.1023/A:1022627411411.

Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20 (3):273–297, 1995b.

Joacir Marques de Oliveira, Eulanda Miranda dos Santos, José Reginaldo Hughes Carvalho, and Leyne Abuim de Vasconcelos Marques. Ensemble of heterogeneous classifiers applied to lithofacies classification using logs from different wells. In *International Joint Conference on Neural Networks, IJCNN*, pages 1–6, Dallas, TX, USA, 2013.

Janez Demsar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006.

Thomas Dietterich and Eun Bae Kong. Machine learning bias, statistical bias, and statistical variance of decision tree algorithms. Technical report, 1995.

Thomas G. Dietterich. Machine-learning research – four current directions. *AI MAGAZINE*, 18:97–136, 1997.

Thomas G. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*, 40(2):139–157, 2000a.

Thomas G. Dietterich. Ensemble methods in machine learning. In *Multiple Classifier SYSTEMS, LBCS-1857*, pages 1–15. Springer, 2000b.

Thomas G. Dietterich and Ghulum Bakiri. Error-correcting output codes: A general method for improving multiclass inductive learning programs. In *Proceeding of AAAI-91*, pages 572–577. AAAI Press, 1991.

Qiang Ding, Qin Ding, and William Perrizo. Decision tree classification of spatial data streams using peano count trees. In *Proceedings of the 2002 ACM symposium on Applied computing*, pages 413–417. ACM, 2002.

Carlos Domingo and Osamu Watanabe. MadaBoost: A modification of AdaBoost. In *Proceedings of the Thirteenth Annual Conference on Computational Learning Theory*, COLT '00, pages 180–189, 2000. ISBN 1-55860-703-X.

Pedro Domingos. A unified bias-variance decomposition. In *Proceedings of 17th International Conference on Machine Learning*, pages 231–238, 2000.

Katti Faceli, Andre Cplf De Carvalho, and Marcilio Cp De Souto. Multi-objective clustering ensemble. *International Journal of Hybrid Intelligent Systems*, 4(3):145–156, 2007.

Sergiy Fefilatyev, Matthew Shreve, Kurt Kramer, Lawrence O. Hall, Dmitry B. Goldgof, Rangachar Kasturi, Kendra Daly, Andrew Remsen, and Horst Bunke. Label-noise reduction with support vector machines. In *ICPR*, pages 3504–3508, 2012.

Manuel Fernández-Delgado, Eva Cernadas, Senén Barro, and Dinani Amorim. Do we need hundreds of classifiers to solve real world classification problems? *Journal of Machine Learning Research*, 15:3133–3181, 2014a.

Manuel Fernández-Delgado, Eva Cernadas, Senén Barro, and Dinani Amorim. Do we need hundreds of classifiers to solve real world classification problems? *Journal of Machine Learning Research*, 15:3133–3181, 2014b.

Antonio Fernández-Baldera and Luis Baumela. Multi-class boosting with asymmetric binary weak-learners. *Pattern Recognition*, 47(5):2080 – 2090, 2014. ISSN 0031-3203.

Eibe Frank and Bernhard Pfahringer. Improving on bagging with input smearing. In *PAKDD*, pages 97–106, 2006.

Benoît Frénay and Michel Verleysen. Classification in the presence of label noise: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 25(5):845–869, 2014.

Yoav Freund. An adaptive version of the boost by majority algorithm. *Machine Learning*, 43(3):293–318, Jun 2001.

Yoav Freund. A more robust boosting algorithm. *arXiv preprint arXiv:0905.2138*, 2009.

Yoav Freund and Robert E. Schapire. Experiments with a new boosting algorithm. In *Proceedings of the Thirteenth International Conference on Machine Learning (ICML 1996)*, pages 148–156, 1996.

Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119 – 139, 1997.

Yoav Freund, Robert Schapire, and N Abe. A short introduction to boosting. *Journal-Japanese Society For Artificial Intelligence*, 14(771-780):1612, 1999.

Mark A Friedl and Carla E Brodley. Decision tree classification of land cover from remotely sensed data. *Remote sensing of environment*, 61(3):399–409, 1997.

Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Additive logistic regression: a statistical view of boosting (With discussion and a rejoinder by the authors). *The Annals of Statistics*, 28(2):337–407, April 2000.

Jerome H. Friedman. On bias, variance, 0/1-loss, and the curse-of-dimensionality. *Data Mining and Knowledge Discovery*, 1(1):55–77, 1997.

Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29:1189–1232, 2000.

Jerome H. Friedman. Stochastic gradient boosting. *Computational Statistics and Data Analysis*, 38(4):367 – 378, 2002. Nonlinear Methods and Data Mining.

Jerome H. Friedman and Peter Hall. On bagging and nonlinear estimation. *Journal of Statistical Planning and Inference*, 137(3):669 – 683, 2007.

G. Fumera and F. Roli. A theoretical and experimental analysis of linear combiners for multiple classifier systems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(6):942–956, June 2005. ISSN 0162-8828. doi: 10.1109/TPAMI.2005.109.

Yunlong Gao and Feng Gao. Edited adaboost by weighted knn. *Neurocomputing*, 73 (16):3079 – 3088, 2010.

Diego García-Gil, Julián Luengo, Salvador García, and Francisco Herrera. Enabling smart data: Noise filtering in big data classification. *Computing Research Repository*, 2017. URL http://arxiv.org/abs/1704.01770.

Robin Genuer, Jean-Michel Poggi, Christine Tuleau-Malot, and Nathalie Villa-Vialaneix. Random forests for big data. *Big Data Research*, 9:28 – 46, 2017. ISSN 2214-5796.

Pierre Geurts, Damien Ernst, and Louis Wehenkel. Extremely randomized trees. *Machine Learning*, 63(1):3–42, 2006. ISSN 1573-0565.

Pall Oskar Gislason, Jon Atli Benediktsson, and Johannes R. Sveinsson. Random forests for land cover classification. *Pattern Recognition Letters*, 27(4):294–300, March 2006. ISSN 0167-8655.

Vanessa Gómez-Verdejo, Manuel Ortega-Moral, Jerónimo Arenas-García, and Aníbal R. Figueiras-Vidal. Boosting by weighting critical and erroneous samples. *Neurocomputing*, 69(7-9):679–685, 2006.

Vanessa Gómez-Verdejo, Jerónimo Arenas-García, and Aníbal R. Figueiras-Vidal. A dynamically adjusted mixed emphasis method for building boosting ensembles. *IEEE Transactions on Neural Networks*, 19(1):3–17, 2008.

Yves Grandvalet. Bagging equalizes influence. *Machine Learning*, 55(3):251–270, 2004.

Adam J. Grove and Dale Schuurmans. Boosting in the limit: Maximizing the margin of learned ensembles. In *Proceedings of the Fifteenth National/Tenth Conference on Artificial Intelligence/Innovative Applications of Artificial Intelligence*, AAAI '98/IAAI '98, pages 692–699, Menlo Park, CA, USA, 1998. American Association for Artificial Intelligence.

Sudipto Guha, Nina Mishra, Gourav Roy, and Okke Schrijvers. Robust random cut forest based anomaly detection on streams. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pages 2712–2721. JMLR.org, 2016.

Ying Guo, Peter L Bartlett, Alexander J Smola, Robert C Williamson, and Jonathan Baxter. Norm-based regularization of boosting. *Technical ReportAustralian National University*, 2002.

Vanessa Gómez-Verdejo, Jerónimo Arenas-García, and Aníbal R. Figueiras-Vidal. Committees of adaboost ensembles with modified emphasis functions. *Neurocomputing*, 73 (7):1289 – 1292, 2010.

Martin T Hagan, Howard B Demuth, Mark H Beale, and Orlando De Jesús. *Neural network design*, volume 20. Pws Pub. Boston, 1996.

Peter Hall and Richard J Samworth. Properties of bagged nearest neighbour classifiers. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(3): 363–379, 2005.

Lars Kai Hansen and Peter Salamon. Neural network ensembles. *IEEE transactions on pattern analysis and machine intelligence*, 12(10):993–1001, 1990.

Mohammad Nazmul Haque, Nasimul Noman, Regina Berretta, and Pablo Moscato. Heterogeneous ensemble combination search using genetic algorithm for class imbalanced data classification. *PloS one*, 11(1):e0146116, 2016.

D. Hernández-Lobato, G. Martínez-Muñoz, and A. Suárez. Statistical instance-based pruning in ensembles of independent classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(2):364–369, 2009.

Daniel Hernández-Lobato, Gonzalo Martínez-Muñoz, and Alberto Suárez. Inference on the prediction of ensembles of infinite size. *Pattern Recognition*, 44(7):1426 – 1434, 2011.

Daniel Hernández-Lobato, Gonzalo Martínez-Muñoz, and Alberto Suárez. On the independence of the individual predictions in parallel randomized ensembles. In *20th European Symposium on Artificial Neural Networks, ESANN 2012, Bruges, Belgium, April 25-27, 2012*, 2012. URL https://www.elen.ucl.ac.be/Proceedings/esann/esannpdf/es2012-19.pdf.

Tin Kam Ho. The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):832–844, 1998.

Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin. A practical guide to support vector classification. Technical report, Department of Computer Science, National Taiwan University, 2003.

W. Hu, W. Hu, and S. Maybank. Adaboost-based algorithm for network intrusion detection. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 38(2):577–583, 2008.

Amaris Jalil, Sierra Hoffman, and Mark Lesser. Predicting temperature at scales relevant for tree growth along elevational gradients in the central appalachians. *Proceedings of the West Virginia Academy of Science*, 89(1), 2017.

Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An Introduction to Statistical Learning: With Applications in R*. Springer Publishing Company, Incorporated, 2014.

Wenxin Jiang. Is regularization unnecessary for boosting? In *Department of Statistics, Northwestern University*, pages 57–64. Morgan Kaufmann, 2001.

Amitava Karmaker and Stephen Kwek. A boosting approach to remove class label noise. *International Journal of Hybrid Intelligent Systems*, 3(3):169–177, 2006.

Manpreet Kaur and Shivani Kang. Market basket analysis: Identify the changing trends of market data using association rule mining. *Procedia computer science*, 85:78–85, 2016.

Daniel Keysers, Thomas Deselaers, Henry A Rowley, Li-Lun Wang, and Victor Carbune. Multi-language online handwriting recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1180–1194, 2017.

A. Khammari, F. Nashashibi, Y. Abramson, and C. Laurgeau. Vehicle detection combining gradient analysis and adaboost classification. In *Proceedings. 2005 IEEE Intelligent Transportation Systems, 2005.*, pages 66–71, 2005.

Taghi M Khoshgoftaar, Shi Zhong, and Vedang Joshi. Enhancing software quality estimation using ensemble-classifier based noise filtering. *Intelligent Data Analysis*, 9(1): 3–27, 2005.

Taghi M. Khoshgoftaar, Jason Van Hulse, and Amir Napolitano. Comparing boosting and bagging techniques with noisy and imbalanced data. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 41(3):552–568, May 2011.

Hyun-Chul Kim, Shaoning Pang, Hong-Mo Je, Daijin Kim, and Sung Yang Bang. Constructing support vector machine ensemble. *Pattern Recognition*, 36(12):2757 – 2767, 2003.

Kyoung-jae Kim and Ingoo Han. Genetic algorithms approach to feature discretization in artificial neural networks for the prediction of stock price index. *Expert systems with Applications*, 19(2):125–132, 2000.

Stefan Knerr, Léon Personnaz, and Gérard Dreyfus. Handwritten digit recognition by neural networks with single-layer training. *IEEE Transactions on Neural Networks*, 3 (6):962–968, 1992.

Ron Kohavi and David Wolpert. Bias plus variance decomposition for zero-one loss functions. In *Machine Learning, Proceedings of the Thirteenth International Conference (ICML '96), Bari, Italy, July 3-6, 1996*, pages 275–283, 1996.

Abba Krieger, Chuan Long, and Abraham Wyner. Boosting noisy data. In *ICML*, pages 274–281, 2001.

Balaji Lakshminarayanan, Daniel M. Roy, and Yee Whye Teh. Mondrian forests: Efficient online random forests. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, pages 3140–3148, Cambridge, MA, USA, 2014. MIT Press.

Louisa Lam and Ching Y. Suen. Optimal combinations of pattern classifiers. *Pattern Recognition Letters*, 16(9):945 – 954, 1995.

Louisa Lam and Ching Y. Suen. Application of majority voting to pattern recognition: an analysis of its behavior and performance. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 27(5):553–568, 1997.

J. J. Lee, P. H. Lee, S. W. Lee, A. Yuille, and C. Koch. Adaboost for text detection in natural scene. In *2011 International Conference on Document Analysis and Recognition*, pages 429–434, 2011.

Andy Liaw and Matthew Wiener. Classification and regression by randomforest. *R News*, 2(3):18–22, 2002a. URL http://CRAN.R-project.org/doc/Rnews/.

Andy Liaw and Matthew Wiener. Classification and regression by randomforest. *R News*, 2(3):18–22, 2002b.

Yong Liu and Xin Yao. Ensemble learning via negative correlation. *Neural Networks*, 12:1399–1404, 1999.

Philip M. Long and Rocco A. Servedio. Random classification noise defeats all convex potential boosters. *Machine Learning*, 78(3):287–304, 2010.

Antonio Loureiro, Luis Torgo, and Carlos Soares. Outlier detection using clustering methods: a data cleaning application. In *Proceedings of KDNet Symposium on Knowledge-based Systems for the Public Sector. Bonn, Germany*, 2004.

Zhenyu Lu, Xindong Wu, and Josh C. Bongard. Active learning through adaptive heterogeneous ensembling. *IEEE Transactions on Knowledge and Data Engineering*, 27(2):368–381, 2015.

Fengjun Lv and Ramakant Nevatia. Recognition and segmentation of 3-d human action using hmm and multi-class adaboost. In Aleš Leonardis, Horst Bischof, and Axel Pinz, editors, *Computer Vision – ECCV 2006*, pages 359–372, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

Jonathan I. Maletic and Andrian Marcus. Data cleansing: Beyond integrity analysis. In *Fifth Conference on Information Quality (IQ 2000)*, pages 200–209, 2000.

Larry M Manevitz and Malik Yousef. One-class svms for document classification. *Journal of machine Learning research*, 2(Dec):139–154, 2001.

Carlos Javier Mantas and Joaquín Abellán. Credal decision trees to classify noisy data sets. In *HAIS*, pages 689–696, 2014.

Gonzalo Martínez-Muñoz and Alberto Suárez. Switching class labels to generate classification ensembles. *Pattern Recognition*, 38(10):1483–1494, 2005.

Gonzalo Martínez-Muñoz and Alberto Suárez. Out-of-bag estimation of the optimal sample size in bagging. *Pattern Recognition*, 43(1):143 – 152, 2010.

Gonzalo Martínez-Muñoz, Aitor Sánchez-Martínez, Daniel Hernández-Lobato, and Alberto Suárez. Building ensembles of neural networks with class-switching. In *Artificial Neural Networks–ICANN 2006*, pages 178–187. Springer, 2006.

Gonzalo Martínez-Muñoz, Aitor Sánchez-Martínez, Daniel Hernández-Lobato, and Alberto Suárez. Class-switching neural network ensembles. *Neurocomputing*, 71(13): 2521–2528, 2008.

Hamed Masnadi-shirazi and Nuno Vasconcelos. On the design of loss functions for classification: theory, robustness to outliers, and savageboost. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 1049–1056. Curran Associates, Inc., 2009.

L Mason, J Baxter, P Bartlett, and M Frean. Boosting algorithms as gradient descent in function space (technical report). *RSISE, Australian National University*, 1999a.

Llew Mason, Peter Bartlett, and Jonathan Baxter. Direct optimization of margins improves generalization in combined classifiers. *Advances in Neural Information Processing Systems*, pages 288–294, 1999b.

Llew Mason, Jonathan Baxter, Peter Bartlett, and Marcus Frean. Boosting algorithms as gradient descent. In *Advances in Neural Information Processing Systems 12*, pages 512–518. MIT Press, 2000.

E. Mayhua-Lopez, V. Gomez-Verdejo, and A. R. Figueiras-Vidal. Real adaboost with gate controlled fusion. *IEEE Transactions on Neural Networks and Learning Systems*, 23(12):2003–2009, 2012.

Efraín Mayhua-López, Vanessa Gómez-Verdejo, and Aníbal R. Figueiras-Vidal. A new boosting design of support vector machine classifiers. *Information Fusion*, 25 (Supplement C):63 – 71, 2015.

Ross A McDonald, David J Hand, and Idris A Eckley. An empirical comparison of three boosting algorithms on real data sets with artificial class noise. In *Multiple Classifier Systems*, pages 35–44. Springer, 2003.

David Mease and Abraham Wyner. Evidence contrary to the statistical view of boosting. *Journal of Machine Learning Research*, 9:131–156, June 2008. ISSN 1532-4435.

Prem Melville and Raymond J. Mooney. Creating diversity in ensembles using artificial data. *Information Fusion*, 6(1):99–111, 2005.

Prem Melville, Nishit Shah, Lilyana Mihalkova, and Raymond J. Mooney. Experiments on ensembles with missing and noisy data. In *Proceedings of the Workshop on Multi Classifier Systems*, pages 293–302. Springer Verlag, 2004.

Tauno Metsalu and Jaak Vilo. Clustvis: a web tool for visualizing clustering of multivariate data using principal component analysis and heatmap. *Nucleic acids research*, 43(W1):W566–W570, 2015.

Qiguang Miao, Ying Cao, Ge Xia, Maoguo Gong, Jiachen Liu, and Jianfeng Song. Rboost: Label noise-robust boosting algorithm based on a nonconvex loss function and the numerically stable base learners. *IEEE Transactions on Neural Networks and Learning Systems*, 27(11):2216–2228, 2016.

Tom M. Mitchell. The need for biases in learning generalizations. In Jude W. Shavlik and Thomas G. Dietterich, editors, *Readings in Machine Learning*, pages 184–191. Morgan Kauffman, 1980. URL http://www.cs.nott.ac.uk/~bsl/G52HPA/articles/Mitchell:80a.pdf. Book published in 1990.

Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of Machine Learning*. The MIT Press, 2012. ISBN 026201825X, 9780262018258.

Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012.

Claude Nadeau and Yoshua Bengio. Inference for the generalization error. *Machine Learning*, 52(3):239–281, Sep 2003.

Loris Nanni, Sheryl Brahnam, Stefano Ghidoni, and Alessandra Lumini. Toward a general-purpose heterogeneous ensemble for pattern classification. *Computational intelligence and neuroscience*, 2015:85, 2015.

Emilie Niaf, Rémi Flamary, Olivier Rouvière, Carole Lartizien, and Stéphane Canu. Kernel-based learning from both qualitative and quantitative labels: Application to prostate cancer diagnosis based on multiparametric mr imaging. *IEEE Transactions on Image Processing*, 23(3):979–991, 2014.

David Opitz and Richard Maclin. Popular ensemble methods: An empirical study. *Journal of Artificial Intelligence Research*, 11:169–198, 1999.

N Oza and K Tumer. Dimensionality reduction through classifier ensembles, computational sciences division. *NASA Ames Research Center, Technical report NASA-ARC-IC-1999-126*, 1999.

Ioannis Partalas, Grigorios Tsoumakas, and Ioannis Vlahavas. Focused ensemble selection: A diversity-based method for greedy ensemble selection. In *Proceedings of the 2008 Conference on ECAI 2008: 18th European Conference on Artificial Intelligence*, pages 117–121. IOS Press, 2008. ISBN 978-1-58603-891-5.

Ioannis Partalas, Grigorios Tsoumakas, and Ioannis Vlahavas. An ensemble uncertainty aware measure for directed hill climbing ensemble pruning. *Machine Learning*, 81(3): 257–282, Dec 2010.

Ioannis Partalas, Grigorios Tsoumakas, and Ioannis Vlahavas. A study on greedy algorithms for ensemble pruning. *Aristotle University of Thessaloniki, Thessaloniki, Greece*, 2012.

Jigar Patel, Sahil Shah, Priyank Thakkar, and Ketan Kotecha. Predicting stock market index using fusion of machine learning techniques. *Expert Systems with Applications*, 42(4):2162–2172, 2015.

Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Edouard Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Andrea Peters and T Hothorn. ipred: improved predictors. R package version 0.8-8, 2009.

Réjean Plamondon and Sargur N Srihari. Online and off-line handwriting recognition: a comprehensive survey. *IEEE Transactions on pattern analysis and machine intelligence*, 22(1):63–84, 2000.

Christian Pölitz and Ralf Schenkel. Robust Ranking Models Using Noisy Feedback. In *Workshop "Information Retrieval Over Query Sessions" (SIR 2012) at ECIR 2012*, pages 1 – 6, 2012.

John Ross Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, March 1986. ISSN 0885-6125.

John Ross Quinlan. Bagging, boosting, and c4.5. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 725–730, 1996.

Gunnar Rätsch. Robust boosting via convex optimization: Theory and applications (doctoral thesis), 2001.

Gunnar Rätsch and Manfred K. Warmuth. Maximizing the margin with boosting. In Jyrki Kivinen and Robert H. Sloan, editors, *COLT*, volume 2375 of *Lecture Notes in Computer Science*, pages 334–350. Springer, 2002.

Gunnar Rätsch and Manfred K Warmuth. Efficient margin maximizing with boosting. *Journal of Machine Learning Research*, 6(Dec):2131–2152, 2005.

Gunnar Rätsch, Takashi Onoda, and Klaus Robert Müller. An improvement of adaboost to avoid overfitting. In *Proceedings of the International Conference on Neural Information Processing*, pages 506–509, Kitakyushu, Japan, 1998.

Lev Reyzin and Robert E. Schapire. How boosting the margin can also boost classifier complexity. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, pages 753–760, New York, NY, USA, 2006. ACM. ISBN 1-59593-383-2.

Marko Robnik-Šikonja. Improving random forests. In Jean-François Boulicaut, Floriana Esposito, Fosca Giannotti, and Dino Pedreschi, editors, *Machine Learning: ECML 2004*, pages 359–370, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg. ISBN 978-3-540-30115-8.

Juan José Rodríguez, Ludmila I. Kuncheva, and Carlos J. Alonso. Rotation forest: A new classifier ensemble method. *IEEE Transactions on Pattern Analysis Machine Intelligence*, 28(10):1619–1630, 2006.

Tijana Ružić and Aleksandra Pižurica. Context-aware patch-based image inpainting using markov random field modeling. *IEEE Transactions on Image Processing*, 24(1): 444–456, 2015.

Maryam Sabzevari, Gonzalo Martínez-Muñoz, and Alberto Suárez. Improving the robustness of bagging with reduced sampling size. In *22th European Symposium on Artificial Neural Networks, ESANN 2014, Bruges, Belgium, April 23-25, 2014*, 2014.

Maryam Sabzevari, Gonzalo Martínez-Muñoz, and Alberto Suárez. Small margin ensembles can be robust to class-label noise. *Neurocomputing*, 160(Supplement C):18 – 33, 2015. ISSN 0925-2312.

Maryam Sabzevari, Gonzalo Martínez-Muñoz, and Alberto Suárez. Pooling homogeneous ensembles to build heterogeneous ensembles. *CoRR*, abs/1802.07877, 2018a. URL http://arxiv.org/abs/1802.07877.

Maryam Sabzevari, Gonzalo Martínez-Muñoz, and Alberto Suárez. Randomization vs optimization in svm ensembles. In Věra Kůrková, Yannis Manolopoulos, Barbara Hammer, Lazaros Iliadis, and Ilias Maglogiannis, editors, *Artificial Neural Networks and Machine Learning – ICANN 2018*, pages 415–421, Cham, 2018b. Springer International Publishing. ISBN 978-3-030-01421-6.

Maryam Sabzevari, Gonzalo Martínez-Muñoz, and Alberto Suárez. A two-stage ensemble method for the detection of class-label noise. *Neurocomputing*, 275:2374 – 2383, 2018c. ISSN 0925-2312.

Maryam Sabzevari, Gonzalo Martínez-Muñoz, and Alberto Suárez. Vote-boosting ensembles. *Pattern Recognition*, 83:119 – 133, 2018d. ISSN 0031-3203.

José A. Sáez, Mikel Galar, Julián Luengo, and Francisco Herrera. Analyzing the presence of noise in multi-class problems: alleviating its influence with the one-vs-one decomposition. *Knowledge and Information Systems*, 38(1):179–206, Jan 2014.

Cullen Schaffer. A conservation law for generalization performance. In William W. Cohen and Haym Hirsh, editors, *Machine Learning Proceedings 1994*, pages 259 – 265. Morgan Kaufmann, San Francisco (CA), 1994. ISBN 978-1-55860-335-6. doi: https://doi.org/10.1016/B978-1-55860-335-6.50039-8.

R. E. Schapire, Y. Freund, P. Bartlett, and W. S. Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. *The Annals of Statistics*, 26(5): 1651–1686, 1998a.

Robert E. Schapire. The strength of weak learnability. *Machine Learning*, 5:197–227, 1990.

Robert E. Schapire and Yoav Freund. *Boosting: Foundations and Algorithms*. The MIT Press, 2012. ISBN 0262017180, 9780262017183.

Robert E Schapire, Yoav Freund, Peter Bartlett, and Wee Sun Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. *Annals of statistics*, pages 1651–1686, 1998b.

Julia Schiffner, Bernd Bischl, and Claus Weihs. Bias-variance analysis of local classification methods. In *Challenges at the Interface of Data Analysis, Computer Science, and Optimization*, pages 49–57. Springer, 2012.

David Sculley and Gabriel M Wachman. Relaxed online svms for spam filtering. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 415–422. ACM, 2007.

Nicola Segata, Enrico Blanzieri, and Pádraig Cunningham. A scalable noise reduction technique for large case-based systems. In *Case-Based Reasoning Research and Development*, pages 328–342. Springer, 2009.

Nicola Segata, Enrico Blanzieri, Sarah Jane Delany, and Pádraig Cunningham. Noise reduction for instance-based learning with a local maximal margin approach. *Journal of Intelligent Information Systems*, 35(2):301–331, 2010.

Chris Seiffert, Taghi M. Khoshgoftaar, Jason V. Hulse, and Amri Napolitano. Resampling or reweighting: A comparison of boosting implementations. In *20th International Conference on Tools with Artificial Intelligence*, pages 445–451, November 2008.

Michael J Shaw, Chandrasekar Subramaniam, Gek Woo Tan, and Michael E Welge. Knowledge management and data mining for marketing. *Decision support systems*, 31(1):127–137, 2001.

J. Shawe-Taylor and N. Cristianini. Robust bounds on generalization from the margin distribution. 1998.

Pannagadatta K. Shivaswamy and Tony Jebara. Variance penalizing adaboost. In John Shawe-Taylor, Richard S. Zemel, Peter L. Bartlett, Fernando C. N. Pereira, and Kilian Q. Weinberger, editors, *NIPS*, pages 1908–1916, 2011.

Borut Sluban and Nada Lavrač. Relating ensemble diversity and performance: A study in class noise detection. *Neurocomputing*, 160(Supplement C):120 – 131, 2015. ISSN 0925-2312.

Borut Sluban, Dragan Gamberger, and Nada Lavrač. Ensemble-based noise detection: noise ranking and visual performance evaluation. *Data Mining and Knowledge Discovery*, 28(2):265–303, Mar 2014.

Alexander J. Smola and Peter J. Bartlett, editors. *Advances in Large Margin Classifiers*. MIT Press, Cambridge, MA, USA, 2000. ISBN 0262194481.

Victor Soto, Alberto Suárez, and Gonzalo Martinez-Muñoz. An urn model for majority voting in classification ensembles. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 4430–4438. Curran Associates, Inc., 2016.

Guillaume Stempfel and Liva Ralaivola. Learning svms from sloppily labeled data. In *ICANN (1)*, pages 884–893, 2009.

Jörg Stork, Ricardo Ramos, Patrick Koch, and Wolfgang Konen. *SVM ensembles are better when different kernel types are combined*, pages 191–201. Springer Berlin Heidelberg, Berlin, Heidelberg, 2015.

Bo Sun, Songcan Chen, Jiandong Wang, and Haiyan Chen. A robust multi-class adaboost algorithm for mislabeled noisy data. *Knowledge-Based Systems*, 102:87 – 102, 2016.

Yijun Sun, Jian Li, and William W. Hager. Two new regularized adaboost algorithms. In *Proceedings of the 2004 International Conference on Machine Learning and Applications - ICMLA 2004, 16-18 December 2004, Louisville, KY, USA.*, pages 41–48, 2004.

Yijun Sun, Sinisa Todorovic, and Jian Li. Reducing the overfitting of adaboost by controlling its data distribution skewness. *International Journal of Pattern Recognition and Artificial Intelligence*, 20(7):1093–1116, 2006.

Grigorios Tsoumakas, Ioannis Katakis, and Ioannis Vlahavas. Effective voting of heterogeneous classifiers. In Jean-François Boulicaut, Floriana Esposito, Fosca Giannotti, and Dino Pedreschi, editors, *Machine Learning: ECML 2004*, pages 465–476, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

Grigorios Tsoumakas, Ioannis Partalas, and Ioannis Vlahavas. *An ensemble pruning primer*, pages 1–13. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

Sergey Tulyakov, Stefan Jaeger, and David Govindaraju, Venuand Doermann. *Machine Learning in Document Analysis and Recognition*, chapter Review of Classifier Combination Methods. Springer Berlin Heidelberg, 2008.

Giorgio Valentini and Thomas G. Dietterich. Bias-variance analysis of support vector machines for the development of svm-based ensemble methods. *Journal of Machine Learning Research*, 5:725–775, 2004. ISSN 1532-4435.

Vladimir Vapnik. Principles of risk minimization for learning theory. In *Advances in Neural Information Processing Systems 4, [NIPS Conference, Denver, Colorado, USA, December 2-5, 1991]*, pages 831–838, 1991.

Vladimir Vapnik. *Statistical learning theory.* Wiley, 1998. ISBN 978-0-471-03003-4.

Vladimir N. Vapnik. *The Nature of Statistical Learning Theory.* Springer-Verlag New York, Inc., second edition, 1999.

Sofie Verbaeten and Anneleen Van Assche. *Ensemble Methods for Noise Elimination in Classification Problems*, pages 317–325. Springer Berlin Heidelberg, Berlin, Heidelberg, 2003.

Ulrike Von Luxburg and Bernhard Schölkopf. Statistical learning theory: Models, concepts, and results. *arXiv preprint arXiv:0810.4752*, 2008.

Chieh-Chih Wang and Ko-Chih Wang. *Hand Posture Recognition Using Adaboost with SIFT for Human Robot Interaction*, pages 317–329. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

Dianhui Wang and Ming Li. Robust stochastic configuration networks with kernel density estimation for uncertain data regression. *Information Sciences*, 412-413 (Supplement C):210 – 222, 2017. ISSN 0020-0255.

Geoffrey I. Webb. Multiboosting: A technique for combining boosting and wagging. In *MACHINE LEARNING*, pages 159–196, 2000.

David P. Williams. Label alteration to improve underwater mine classification. *IEEE Geoscience and Remote Sensing Letters*, 8(3):488–492, 2011.

Terry Windeatt. Accuracy/diversity and ensemble MLP classifier design. *IEEE Transactions on Neural Networks*, 17(5):1194–1211, 2006.

Ian H. Witten, Eibe Frank, and Mark A. Hall. *Data Mining: Practical Machine Learning Tools and Techniques, 3rd Edition.* Morgan Kaufmann, 2011.

David H. Wolpert. Stacked generalization. *Neural Networks*, 5:241–259, 1992.

David H Wolpert, William G Macready, et al. No free lunch theorems for search. Technical report, Technical Report SFI-TR-95-02-010, Santa Fe Institute, 1995.

Jonathan Young, John Ashburner, and Sebastien Ourselin. Wrapper methods to correct mislabelled training data. In *Proceedings of the 2013 International Workshop on Pattern Recognition in Neuroimaging*, PRNI '13, pages 170–173. IEEE Computer Society, 2013. ISBN 978-0-7695-5061-9.

Reda Younsi and Anthony Bagnall. Ensembles of random sphere cover classifiers. *Pattern Recognition*, 49:213 – 225, 2016.

Zhiwen Yu, Daxing Wang, Jane You, Hau-San Wong, Si Wu, Jun Zhang, and Guoqiang Han. Progressive subspace ensemble learning. *Pattern Recognition*, 60:692 – 705, 2016. ISSN 0031-3203.

Faisal Zaman and Hideo Hirose. Effect of subsampling rate on subbagging and related ensembles of stable classifiers. In *Pattern Recognition and Machine Intelligence*, pages 44–49. Springer, 2009.

Shi Zhong, Wei Tang, and Taghi M Khoshgoftaar. Boosted noise filters for identifying mislabeled data. Technical report, Technical report, Department of computer science and engineering, Florida Atlantic University, 2005.

Bing Zhou, Yiyu Yao, and Jigang Luo. Cost-sensitive three-way email spam filtering. *Journal of Intelligent Information Systems*, 42(1):19–45, 2014.

Yong Zhou, Youwen Li, and Shixiong Xia. An improved knn text classification algorithm based on clustering. *Journal of computers*, 4(3):230–237, 2009.

X. Zhu, Chengyi Bao, and Weidong Qiu. Bagging very weak learners with lazy local learning. In *2008 19th International Conference on Pattern Recognition*, pages 1–4, Dec 2008.

Xingquan Zhu and Xindong Wu. Class noise vs. attribute noise: A quantitative study. *Artificial Intelligence Review*, 22(3):177–210, 2004a.

Xingquan Zhu and Xindong Wu. Class noise vs. attribute noise: A quantitative study. *Artificial Intelligence Review*, 22(3):177–210, Nov 2004b. ISSN 1573-7462.

Xingquan Zhu, Xindong Wu, and Qijun Chen. Eliminating class noise in large datasets. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pages 920–927, 2003.